Universidade de Aveiro Departamento de Engenharia Mecânica
2007

Nuno Filipe dos Santos Beça Pereira

Serviços *Web* e arquitecturas de serviços de *Broker* para empresas ágeis e virtuais.

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Mestrado Integrado em Engenharia Mecânica, realizada sob a orientação científica do Prof. Doutor José Paulo Oliveira Santos, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro e co-orientação do Prof. Doutor Goran Putnik, Professor Associado com Agregação do Departamento de Produção e Sistemas da Universidade do Minho.

Dedicatória A Meus Pais por sempre terem acreditado em mim…

o júri

Presidente	Professor Doutor Vítor Manuel Ferreira dos Santos Professor Associado, Departamento de Engenharia Mecânica, Universidade de Aveiro
Arguente	Professora Doutora Maria Teresa Braga Valente de Almeida Restivo Investigadora Principal; Secção de Automação, Instrumentação, e Controlo; Departamento de Engenharia Mecânica e Gestão Industrial; Faculdade de Engenharia da Universidade do Porto
Arguente	Professor Doutor Rui António da Silva Moreira Professor Auxiliar, Departamento de Engenharia Mecânica, Universidade de Aveiro
Orientador	Prof. Doutor José Paulo Oliveira Santos Professor Auxiliar, Departamento de Engenharia Mecânica, Universidade de Aveiro
Arguente	Professor Doutor Goran Putnik Professor Associado com Agregação; Departamento de Produção e Sistemas; Centro de Engenharia de Sistemas de Produção da Universidade do Minho

Agradecimentos

À minha namorada Sílvia pelo apoio e pela compreensão, ao meu irmão Pedro pela ajuda e os conselhos sempre sábios, aos meus pais por todo apoio dado, ao Professor Dr. José Paulo Santos e Professor Dr. Goran Putnik por todo o conhecimento transmitido e ajuda dada e aos meus colegas de mestrado pelos bons momentos de trabalho passados.

Agradeço a todos os que, de uma maneira ou de outra, me deram o apoio e carinho especiais nesta fase da minha vida.

palavras-chave

Empresas Ágeis e Virtuais, Web Services, SOAP, UDDI, WSDL, Broker, Comércio Electrónico

resumo

Desde os finais do século XX o aumento exponencial da competitividade das empresas a nível global tornou insustentável o tipo de indústria até aí existente. Esta indústria caracterizava-se por ser pouco flexível, ou seja, as empresas não tinham capacidade de se adaptar rapidamente às alterações de mercado nem de partilhar recursos para beneficio de todos. Houve então a necessidade de procurar novas tecnologias e ideologias (sistemas flexíveis de produção) para essas empresas se tornarem cada dia mais eficientes, eficazes e flexíveis. Foi também necessário alterarem os meios de divulgação dos seus produtos e serviços de forma a poderem ser contratadas ou poderem subcontratar serviços a outras empresas. Meios de divulgação como o telefone, fax e carta tornaram-se obsoletos na localização de Fornecedores, solicitação de orçamentos e de realização encomendas.

Como a oferta de produtos e serviços aumentou exponencialmente e, sendo as margens de lucro cada vez menores, surgiu o desafio de como conseguir efectuar uma procura pormenorizada que seleccionasse os melhores Fornecedores. Enquanto antigamente se pediam orçamentos a algumas empresas, de uma mesma zona geográfica, hoje em dia isso já quase não acontece, dado que são procuradas as empresas que fazem melhor, mais barato e mais rápido, não importando se essas ficam ou não na mesma zona geográfica.

Hoje em dia a Internet afirma-se como um meio privilegiado e indispensável para as empresas trocarem e disponibilizarem de uma forma rápida e eficaz todo o tipo de dados. No entanto constata-se que empresas ditas “familiares” ainda não aderiram a esta forma de divulgação. Neste sentido, este trabalho propõe-se a comparar arquitecturas de mercado baseadas em serviços Web para fornecer aos Clientes uma forma simples, rápida e automática de adquirir produtos através da Web, com ou sem intervenção humana.

Para a realização deste estudo e com base em requisitos predefinidos, serão propostas arquitecturas para representar transacções via Web e será implementado um Broker que seja capaz de localizar automaticamente as entidades capazes de fornecer um determinado produto ou serviço e seleccionar qual o melhor tendo em conta critérios definidos pelo Cliente. Este Broker terá duas funcionalidade, uma que será parcialmente controlada pelo Cliente e outra que será totalmente autónoma e não terá intervenção humana.

De uma forma genérica o Broker tem como objectivo encontrar, em qualquer parte do mundo, uma entidade que possa ser subcontratada (Fornecedor) e que produza um determinado Produto / Serviço ao mais baixo preço e/ou o mais rapidamente possível utilizando a Web como meio de transmissão são utilizadas as linguagens de programação Visual C# 2005 e HTML. Os serviços Web propostos são baseados nas especificações SOAP, WSDL e UDDI.

keywords

Agile and Virtual Enterprise, Web Services, SOAP, UDDI, WSDL, Broker, Ecommerce

abstract

Since the end of the twentieth century, due to the exponential increase in the competitiveness of businesses globally, it became untenable to the type of industry there, which was an industry lacks flexibility, that is, companies had no capacity to adapt quickly to market changes. There was then the need to seek new technologies and ideologies (flexible systems of production) every day to become more efficient, effective and flexible. There was also a need to change the way of disclosing their products and services to be contracted or can subcontract services to other companies. Means of localization like the phone, fax and letter have become obsolete to locate suppliers, request budgets and realize orders.

As the supply of goods and services has increased exponentially, and increasing the profit margins are smaller, so there was a need for a detailed search for the so select the best suppliers. While past budgets if asked to half a dozen companies, and they are in the same geographical area, today this has hardly happens because companies increasingly seek whom do better, cheaper and faster, no matter if it is in the same geographical area or not.

The Internet nowadays already been stated as a means privileged and indispensable for businesses to exchange and provide a rapid and effective any type of data, only the companies said "family" is not yet acceded to this form of disclosure. Therefore, this work it is proposed to compare architectures of the market based on Web Services to provide customers a simple, fast and automatic buy product through the Web with or without human intervention. . To carry out this study and based on predefined requirements, there will propose architectures and implement a Broker, which is able to automatically locate the entities capable of providing a particular product or service and select what better taking into account criteria set by Customer . This will have two Broker functionality, one that will be partially controlled by Customer and another that will be completely autonomous and virtually without human intervention.

As a general the Broker, aims to find in the world, an entity that can be subcontracted (Provider), which produces a product / service at the lowest price and / or as soon as possible. With the Web as a means of transmission, are used programming language Visual C # 2005 and HTML. The services offered are based on Web specifications SOAP, WSDL and UDDI.

Índice:

Índice de Figuras:

Índice de Tabelas:

Definições

Ao longo desta dissertação serão utilizadas algumas palavras-chave: umas que se inserem num contexto mais geral e que aparecem destacadas a **bold**, como é o caso das palavras Fornecedor, Cliente e Produto ou Serviço. A *itálico* aparecem as palavras Inglesas. As palavras que surgem num contexto mais objectivo aparecem iniciadas sempre com letra maiúscula e a *itálico* .

Passa-se a explicar cada definição de forma sucinta:

Arquitectura – conjunto de serviços organizados em vários componentes que estão ligados entre si, interagindo.
A *Arquitectura* está dividida em duas partes, a parte Física e a parte Lógica. A parte Física representa as interligações entre os vários elementos e a parte Lógica representa a estrutura do software e está estruturada por seis campos:
processos de negócio (*business processes*);
directório de recursos (*Brokerage services*) ferramentas de procura, selecção, negociação electrónica, segurança e autenticação;
middleware, protocolos: UDDI, SOAP, ... ;
plataforma de aplicação (*applicational plataform*);
armazenamento de informação (*datarepository*), Base de Dados local, remota ou distribuída;
estrutura física (*physical infrasturute*) .

- *Computação Ubíqua* – consiste em desenvolver ambientes informáticos onde a introdução de novas tecnologias no quotidiano ocorra de modo natural e transparente.
- *Sistema de Produção e Empresas Ubíquas* – trata-se de um sistema de produção dinâmico e automaticamente reconfigurável constituído por um conjunto de dispositivos inteligentes e flexíveis, interligados para execução de novas tarefas podendo ser controlado a partir de qualquer parte do mundo. Como exemplo deste dispositivo inteligente temos uma célula robotizada de soldadura, onde os sensores detectam presença da peça a soldar numa determinada posição dentro do seu raio de acção. Esta informação é passada para um outro dispositivo de nível superior dessa célula (por exemplo de um braço do robot de soldadura) que em seguida pode passar a informação para a garra dos eléctrodos e deste para outros dispositivos da rede. Assim tem-se um sistema flexível de produção.
- *Regulador* – aplicação informática responsável por zelar pela concretização da transacção e por monitorizar a transacção, mas sem entrar na cadeia de mensagens. Para poder entrar na transacção todos os elementos têm de pertencer à sua Base de Dados.

- ***Broker*** – aplicação informática, residente num endereço URL, que disponibiliza determinados serviços capazes de descobrir **Fornecedores** que podem fornecer um determinado **produto / serviço**, seleccionar as melhores propostas, realizar encomendas e fazer os seus seguimentos.
- **Clientes** – entidades que acedem ao endereço URL do Broker para adquirirem um determinado **produto / serviço**.
- **Produtos** – bens ou objectos disponibilizados pelos **Fornecedores**.
- **Fornecedores** – entidades que, através dos seus recursos e tecnologias, transformam a matéria-prima em **produtos** solicitados pelos **Clientes**.
- *Serviços Web* – funções disponibilizadas pelos **Fornecedores** que permitem a interacção com as restantes entidades envolvidas (***Broker***, **Cliente**).

Todas as citações de outros autores, assim como textos transcritos de outros trabalhos ou sites da *Internet* são ainda formatadas a itálico e entre aspas.

São utilizados também nesta dissertação as chamadas siglas e acrónimos que são escritos em letra maiúscula seguidos, na sua primeira utilização, do respectivo significado entre parênteses.

Exemplo: SOAP (*Simple Object Access Protoco*l)

Ao longo desta dissertação são usados dois termos semelhantes, *Web Services* e Serviços *Web*, que poderão causar alguma confusão no decorrer da leitura. Para tal passa-se a explicar cada um dos termos:

Web Services - refere-se à tecnologia de suporte utilizada;

Serviços *Web* - refere-se aos serviços propostos pelo **Fornecedor** e que o ***Broker*** poderá invocar, recorrendo ou não, aos *WebServices*.

CAPÍTULO 1

1.1 Introdução

O facto de haver a possibilidade de ter um sistema informático capaz de disponibilizar, partilhar, divulgar informação e tomar decisões, num determinado local e momento foi, e continua a ser, um factor determinante na diferenciação entre pessoas, entre organizações, entre empresas e entre países. No acesso e troca de informação de uma forma rápida e segura são reconhecidas vantagens e são factores que contribuem muito para o desenvolvimento em todas as áreas de conhecimento, produção e negócio.

O aparecimento do computador[1] (1946) e o desenvolvimento dos sistemas informáticos tiveram um papel preponderante no desenvolvimento tecnológico e científico do século XX. Com o inicio da comercialização dos computadores (1951) e com o sucessivo avanço tecnológico, na década de 80 / 90 passou a ser possível a cada pessoa ter o seu próprio computador e assim poder instalar as suas próprias aplicações e gerir os seus próprios dados.

Mais tarde, com o aparecimento da *Internet*, passou a ser imperativo a partilha desses mesmos dados com outros utilizadores através de bases de dados remotas, localizadas em servidores centrais. Os computadores executavam as aplicações necessárias para aceder, modificar ou ainda manipular os dados pessoais que apenas serviam a um utilizador em particular.

O simples uso do computador para fins pessoais foi-se transformando, de forma rápida e entusiástica, numa revolução tecnológica que até aos dias de hoje não sofreu nenhum retrocesso.

Hoje, pode dizer-se que todas as empresas se encontram informatizadas, isto é, têm computadores, redes internas, ligações à *Internet* e utilizam inúmeros dispositivos informáticos com o objectivo de melhorarem a sua produção, controlo de stocks, agilizarem os seus processos de fabrico e também torna-los mais inteligentes. Esta inteligência é o resultado de um elevado número de decisões, tomadas pelos dispositivos nas suas áreas de trabalho e que são guardadas em Base de Dados, para posteriormente serem usadas para ajudarem a tomar decisões. Ajudou também na melhoria dos meios de divulgação e disponibilização dos seus **produtos** ou **serviços** na maior rede de informação o que levou a que cada vez mais se use a subcontratação para a realização de tarefas que antigamente eram realizadas dentro das empresas. Portanto, hoje torna-se muito mais fácil e simples mandar fazer fora do que na própria empresa, isto porque, permite conseguir uma maior rentabilidade das máquinas e são necessários menos funcionários e stocks.
A *Internet*, através do chamado comércio electrónico, permite que estas empresas assegurem um lugar competitivo no mercado onde actuam.

---

[1] O primeiro computador do mundo foi o **ENIAC** (*Electronic Numerical Integrator and Computer*), feito pelo Prof. John Mauchly, conjuntamente com o Prof. J. Presper Eckert. Mauchly e o Eckert em 1943.

Existem muitas definições para comércio electrónico, das quais se destacam:

A definição da ANACOM (Autoridade Nacional de Comunicações) segundo a qual "A disponibilização da tecnologia *World Wide Web*, durante os anos 90, permitiu à grande parte do tecido empresarial reequacionar as estratégias de actuação no mercado, provocando alterações profundas no ambiente negocial tradicional, designadamente no modo de relacionamento entre **Clientes** e **Fornecedores**. Esta alteração deu origem a uma nova forma de vender e comprar – o Comércio Electrónico – que se tem convertido num factor fundamental de competitividade e num fortíssimo indutor de produtividade para a generalidade das empresas". [ANACOM 2004]

Existe ainda a definição de SPI (Sociedade Portuguesa de Inovação) segundo a qual: "Em termos simplistas, Comércio Electrónico poder-se-ia definir como realizar negócios electronicamente". Uma definição mais elaborada poderia resumir-se a "qualquer tipo de transacção comercial, em que as partes envolvidas interajam electronicamente e não através de trocas ou contactos físicos". [SPI 2000]

1.2 O Problema

A necessidade de modificar o modo de produção rígido, fixo e de difícil reconfiguração de grande parte das empresas da actualidade, leva a que cada vez mais sejam precisos sistemas flexíveis de produção e que hajam cada vez mais boas relações comerciais entre empresas do mesmo ramo. É impensável nos dias que correm que uma empresa tenha grandes stocks de peças, tenha sistemas fixos de produção, porque cada vez mais são precisas empresas que tenham capacidade de se adaptar às circunstancias, ter boa carteira de **Fornecedores**, conseguir subcontratar serviços externos quando precisar, em vez de os ter internamente, como por exemplo o serviço de CAD, cada vez mais as empresas têm um departamento de Desenho 3D pequeno e no caso de precisarem de trabalho extra subcontratam o serviço de outra empresa.

Comércio electrónico pode ser descrito como a actividade de compra e venda de bens ou serviços apoiados, de alguma forma, por uma infra-estrutura tecnológica computacional distribuída.

Com o aparecimento e utilização da *Internet*, este conceito começou a ser mais utilizado, aceite e divulgado, substituindo ou tornando-se numa alternativa às tecnologias anteriormente existentes (telefone, fax, cartas, etc.). Com o crescente número de pessoas e empresas a utilizar a *Internet* para este tipo de negócio, houve a necessidade da *Internet* se adaptar, sofrendo constantes evoluções de forma a dar resposta às necessidades do comércio electrónico. O aumento das velocidades de acesso e a possibilidade de fazer pagamentos através da *Web,* tornou-a num sistema ainda mais aberto e propício à sua utilização e expansão.
Assim, a *Web* afirma-se cada vez mais como um meio rápido e eficaz de disponibilizar, adquirir e trocar informações em qualquer momento e em qualquer parte do mundo.

Cada vez mais as empresas apostam neste meio de informação para se darem a conhecer, assim como aos seus **produtos**, com o intuito de aumentarem os seus volumes de negócios, tornando-se mais competitivas nos mercados onde se inserem. No entanto a competitividade destas empresas não depende apenas do facto de serem, ou não, conhecidas por outras empresas ou pessoas. A maior ou menor rapidez e facilidade em descobrir e/ou subcontratar bens ou serviços a outras empresas, a forma como se adaptam ao imprevisto, como por exemplo a falhas na entrega por parte do **Fornecedor**, a selecção da melhor proposta ou do melhor **Fornecedor**, não só em termos de fornecimento de **produto** mas também em termos de tempos de entrega, confidencialidade e outros aspectos que podem influenciar negativamente a transacção. Estes aspectos podem ser um meio de melhorar substancialmente a satisfação quer dos **Clientes** mas também dos **Fornecedores**, não só porque se conseguem menores custos de aquisição mas também porque se conseguem transacções mais seguras, ou seja, os **Fornecedores** têm a garantia de receber o pagamento pelo serviço prestado e o **Cliente** tem a garantia da entrega do **produto** na data pretendida e nas condições acordadas e de confidencialidade.

Outro aspecto importante, prende-se com custos de localização e negociação de serviços que são seguramente um dos economicamente mais penalizantes. Como “tempo é dinheiro”, seria desejável reduzir ao máximo os custos de localização e negociação. O tempo despendido na pesquisa de empresas, na redacção de documentos e pedidos de orçamentos, na escrita de propostas, na formalização de encomendas, nas ordens para os bancos, nas confirmações e demais procedimentos burocráticos e financeiros são factores que contribuem significativamente para estes custos.

1.3 Objectivos

Este trabalho tem como objectivo o estudo de arquitecturas de serviços de ***Broker*** para empresas ágeis e virtuais que possam não só reduzir o tempo e/ou os custos da subcontratação (Descoberta de **Fornecedores**, Orçamentação, Selecção, Realização e Acompanhamento de Encomendas) mas também garantir que a transacção corre bem, garantir a confidencialidade dos intervenientes, o respeito pelas condições quer de pagamento mas também prazos, condições de entrega e garantir que haja capacidade de se adaptar aos imprevistos, como seja o caso de ruptura de stocks do fornecedor, avarias ou outras implicações que façam com que não se possa cumprir o prazo e condições de entrega. Para isso recorrendo às novas tecnologias da informação, entre elas *Web Services* e Serviços *Web*.

Idealmente seria desejável testar todas as cinco *Arquitecturas* que serão descritas mais à frente, implementando cada uma delas e para isso recorrer a entidades externas (***Broker*** e Regulador) de forma rápida, automática e autónoma. Descobrir o maior número de empresas na *Web*, contactá-las individualmente pedindo-lhes orçamentos para a aquisição de um determinado

produto ou **serviço**, em seguida analisar todas as propostas recebidas e no final apresentar a melhor proposta (prazo de entrega mais reduzido e/ou a solução mais económica) se o **Cliente** assim o pretendesse. O **Cliente** teria a possibilidade de escolher se o processo de encomenda do bem ou serviço fosse totalmente automático ou parcialmente ou ainda totalmente manual. O **Cliente** teria a cada instante acesso à informação relativa ao estado da encomenda (em produção, concluída ou já foi enviada).
No final destas operações, o **Cliente** que pretendesse adquirir um determinado **produto** ou **serviço**, não deveria colocar nenhuma das seguintes questões:

- Terão sido contactadas todas as empresas capazes de fornecer uma proposta de orçamento para a aquisição deste bem ou serviço?
- Já terei recebido as propostas de orçamentos de todas as empresas contactadas?
- Será que vou encontrar Fornecedores capazes para fornecerem o produto que pretendo?
- O Fornecedor terá capacidade de me fornecer o produto mesmo que aconteçam imprevistos?
- A minha empresa será capaz de dar resposta a encomendas com prazos de entrega reduzidos?
- Serei capaz de garantir prazos de entrega, mesmo que me apareçam imprevistos?
- Terei de fazer grandes investimentos caso apareçam encomendas para as quais eu não tenho meios de produção suficientes?
- Será que é melhor ter uma grande quantidade de **produtos** em stock, para caso apareça uma grande encomenda?
- A minha empresa será capaz de se adaptar rapidamente a imprevistos de produção?

No entanto, por uma questão de tempo e disponibilidade serão implementadas e estudadas apenas as *arquitecturas* n.º 4 e 5, por serem as mais abrangentes e assim se poder tirar melhores conclusões.

1.4 Situação Actual

Os **Clientes** quando pretendem adquirir um bem ou serviço, recorrem frequentemente a uma lista conhecida onde se encontram as localizações e contactos de todos os **Fornecedores** que disponibilizam os bens ou serviços que pretendidos. Por isso, geralmente são contactados um número reduzido de **Fornecedores** e nada garante que esses mesmo **Fornecedores** sejam fidedignos e que vão cumprir o estipulado.

Quando as empresas pretendem contratar novos **serviços** (**Produtos**) podem surgir dificuldades de negociação traduzindo-se normalmente em orçamentos economicamente desfavoráveis.
Outra maneira cada vez mais usada é através da *Web*. O **Cliente** pode descobrir informações sobre os **Fornecedores** a partir de palavras-chave, utilizando motores de busca tais como GOOGLE, YAHOO, entre outros. Com base nos resultados da pesquisa, o **Cliente** pode aceder às páginas *Web* dos potenciais **Fornecedores**, e aí pesquisar os **produtos** que cada uma disponibiliza. De acordo com as informações encontradas o **Cliente** pode entrar em contacto com os **Fornecedores** para solicitar orçamentos, podendo ser este contacto por telefone, faxes e ou e-mail's. A recepção dos orçamentos é um processo que pode demorar vários dias, semanas e o **Cliente** pode até nunca receber todos os orçamentos solicitados. Vai caber ao próprio **Cliente** decidir se já tem orçamentos suficientes ou se pretende esperar mais.
O **Cliente** de forma manual compara os orçamentos recebidos e ordena-os com base nos custos e/ou tempos de entrega. O passo seguinte é entrar em contacto com o **Fornecedor** para encomendar o serviço e, posteriormente esperar, e "rezar" que chegue dentro do prazo estipulado. Só quando receber a encomenda é que o **Cliente** vai poder verificar se era ou não o que estava acordado. Durante este tempo o **Cliente** para saber o estado da evolução da sua encomenda terá de contactar o **Fornecedor** .
Mesmo que o **Cliente** tenha acesso à *Internet* para fazer a procura de **Fornecedores**, o tempo que se perde desde a procura, envio de pedidos de orçamentos, recepção dos orçamentos, formalização da encomenda, pagamento e recepção da encomenda pode passar bastante tempo e implicar custos elevados, para além disso, o resultado da pesquisa é limitado a um número reduzido de **Fornecedores**.

1.5 Solução Proposta

Este trabalho propõe comparar e implementar diferentes *Arquitecturas* conceptuais que para Serviços flexíveis, virtuais, dinâmicos de subcontratação e reconfiguráveis que permitam descobrir e subcontratar bens ou serviços em todo o mundo através da *Web* garantindo sempre a boa concretização da transacção, localizar, seleccionar e encomendar de forma automática e autónoma. Para isso, recorrer-se à nova tecnologia da informação e comunicação. Um ***Broker*** que irá implementar esses Serviços *Web* e a um *Regulador* que monitorizará toda transacção, garantindo assim que a transacção ocorra como pretendido.

Neste estudo abordar-se-á:
A apresentação de várias *arquitecturas* conceptuais que propõe a existência de um *Regulador*, de um ***Broker***, e de serviços *Web* dos **Fornecedores**.
Uma implementação possível desse ***Broker*** e dos serviços *Web* dos **Fornecedores** para avaliar as *arquitecturas* conceptuais, os serviços propostos e os seus desempenhos.

Os serviços *Web* propostos neste trabalho, são baseados nas especificações SOAP (*Simple Object Access Protocol*), WSDL (*Web Service Description Language*) e UDDI (*Universal Description Discovery and Integration*) e no protocolo HTTP (*HiperText Transfer Protocol*):

HTTP – protocolo de comunicação que permitirá a transferência de informação pela *Web* entre servidores *WEB* e browsers a nível mundial.

XML (*Extensible Markup Language*) – linguagem que permitirá descrever, armazenar, trocar e manipular estruturas de dados, facilitando a sua interpretação assim como a criação de novas aplicações de manipulação e visualização dos dados através da *Web*.

SOAP – O SOAP é o protocolo simples do acesso do objecto. SOAP é baseado em XML e descreve um formato de mensagens para comunicação entre máquinas. É a especificação responsável pela invocação de aplicações remotas e mecanismos de transporte via HTTP, através de *RPC* (*Remote Procedure Call*) ou trocas de mensagens em sistemas onde a plataforma ou linguagem de programação não se coloca como parâmetro importante, garantindo desta forma a interoperabilidade e comunicação entre sistemas distribuídos.

WSDL – especificação, baseada no XML, utilizada para descrever os serviços disponibilizados pelos Serviços *Web*. Neste caso em particular será a linguagem que permitirá a cada **Fornecedor**, neste contexto, descrever de forma detalhada os Serviços *Web* que disponibiliza, as operações que pode realizar, o formato das mensagens a processar, os protocolos que suporta e a sua localização.

UDDI – é uma especificação que define um serviço de registro para *Web Services*.

Servidor UDDI – local na *Web* onde se encontram registados Serviços *Web* (funciona com a mesma filosofia das Páginas Amarelas). Neste caso em particular, é a especificação que permitirá ao ***Broker*** proposto descobrir a nível mundial os **Fornecedores** que podem ser seleccionados para realizar os serviços pretendidos. É composto por um conjunto de registos e fornece informações sobre negócios ou entidades aí registadas.

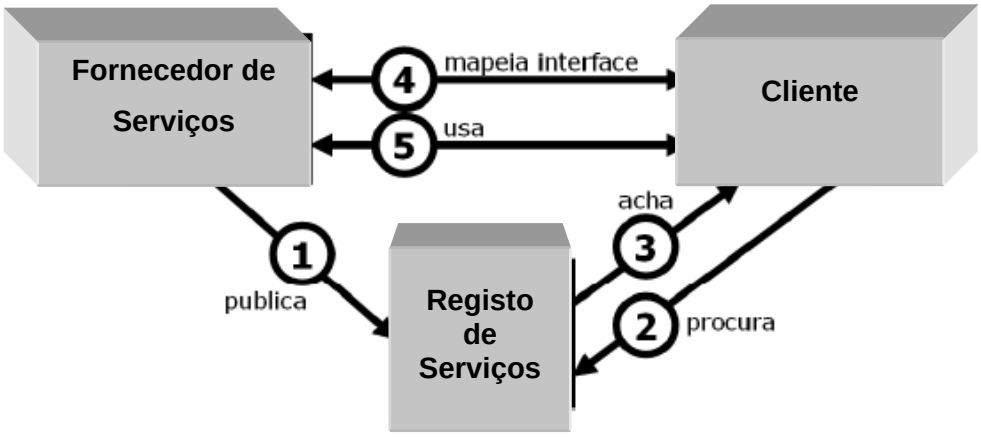

Figura 1 – *Arquitectura* dos *Web Services*.

1.6 Organização da dissertação

No capítulo 2 são descritas as tecnologias e especificações existentes e a forma como elas podem ser aplicadas no âmbito desta dissertação. São ainda referidas as soluções encontradas noutras teses e trabalhos realizados.
No capítulo 3 são apresentadas as várias arquitecturas analisadas, a solução proposta e a sua implementação. São também apresentadas as entidades envolvidas, a forma como foram implementadas e a forma como interagem entre si, e os serviços propostos definidos na linguagem WSDL.
No capítulo 4 é apresentada uma comparação qualitativa das cinco (5) arquitecturas estudadas e apresenta-se no final um tabela comparativa das várias arquitecturas.
No capítulo 5 é apresentada a solução proposta, explica-se em pormenor o que foi realizado, bem como as tecnologias de suporte utilizadas e no final são apresentados os resultados obtidos com a realização do trabalho.
No capítulo 6 fala-se das conclusões obtidas, considerações gerais e trabalhos que ficaram em aberto para realização num futuro próximo.
Por fim no capítulo 7 são apresentadas as referencias que foram para a realização deste trabalho.

CAPITULO 2

2 ESTADO DA ARTE

2.1 Tecnologias de Suporte

Neste capítulo irá abordar-se as tecnologias de suporte aos sistemas distribuídos e a forma como serão utilizadas na solução proposta nesta dissertação. Entende-se por um sistema computacional distribuído uma rede informática onde existem vários computadores e outros recursos que podem e pretendem partilhar as suas aplicações ou serviços com outros computadores existentes nessa mesma rede. Assim, pode-se dizer que um determinado computador pode comportar-se, por um lado, como cliente quando requer serviços a outro computador da rede e, por outro lado, como servidor quando fornece os serviços a outros computadores como por exemplo servidor de correio electrónico.

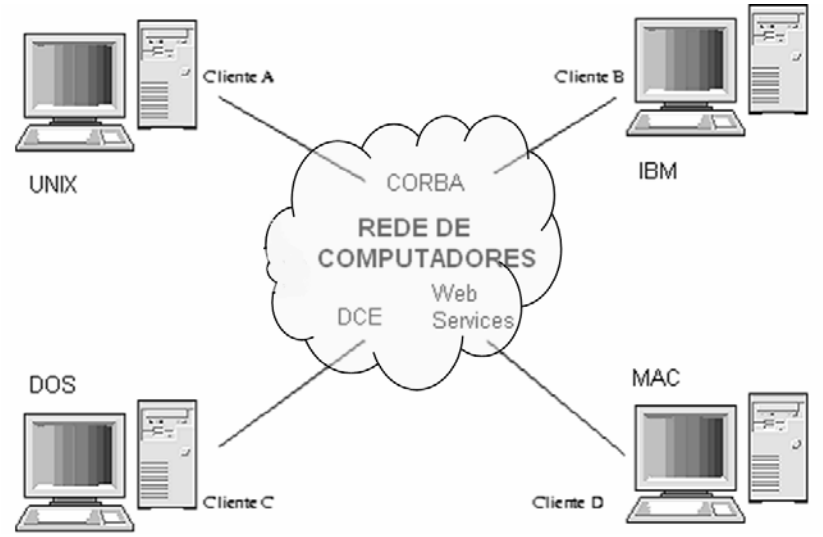

Figura 2 – Exemplo de um sistema computacional distribuído.

Assim pode-se dizer que a palavra distribuído significa que estes computadores e outros recursos da rede se encontram geograficamente descentralizados, cada pode ter os seus serviços e fazer de servidor para esse serviço e requerer serviços que estão noutras máquinas, comportando-se assim como cliente desse serviço.

Existem várias motivações para a utilização de sistemas computacionais distribuídos em alternativa aos sistemas computacionais centralizados. Por um lado a utilização de sistemas computacionais distribuídos exige maiores cuidados a nível da segurança, confiança e desempenho da rede de comunicação. Não obstante, a utilização destes sistemas permite a

partilha de recursos e informações entre subsistemas, o que se pode traduzir em significativos ganhos económicos e tecnológicos. [Tanenbaum 1992]

Os programadores ao criarem aplicações distribuídas, pretendem que estas possam interagir com outras existentes noutros computadores. Para isso é conveniente que não seja fazer alterações de base ou até mesmo refazer as aplicações, daí os programadores devem utilizar as especificações, normas e/ou até mesmo protocolos (Tecnologias de Suporte) de forma a permitirem uma mais rápida e transparente implementação e interacção com outras aplicações já existentes na rede distribuída. Estas tecnologias de suporte devem ser capazes de localizar servidores, identificar os seus serviços e fornecer mecanismos de interacção com eles.

Na última década, diversas tecnologias de suporte foram desenvolvidas com o propósito de facilitar o desenvolvimento de aplicações, abstraindo os detalhes da comunicação, invocação de procedimentos e serviços, nomes e localizações. O termo *middleware* pode ser apresentado como sendo a camada intermédia de software que possui os mecanismos que possibilitam a comunicação entre aplicações distribuídas, podendo ser sistemas operativos diferentes, diminuindo assim a complexidade e a heterogeneidade dos sistemas existentes (Figura 3).

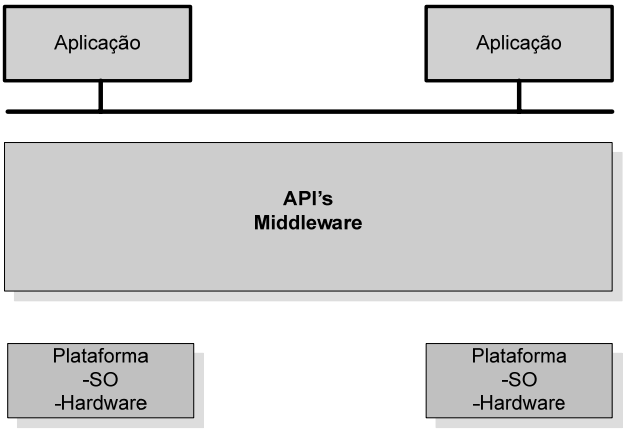

Figura 3 – Camada *middleware* num sistema distribuído [Fraunhofer 2006]

Tanto o DCE (*Distributed Computing Environment*), como o CORBA (*Common Object Request **Broker***), como os Serviços *Web* são tecnologias *middleware* que suportam a construção e implementação cliente/servidor em sistemas computacionais distribuídos. Ao ter disponível e ao utilizar-se o *middleware* verifica-se uma considerável redução da complexidade no desenvolvimento de novos sistemas, porque já está garantida a interoperacionalidade entre os vários sistemas operativos, aplicações e plataformas computacionais.

2.1.1 DCE - Distributed Computing Environment

O DCE propõe um conjunto de serviços que pretende facilitar o desenvolvimento, a flexibilidade e a operacionalidade de aplicações informáticas em sistemas distribuídos. Várias implementações deste ambiente distribuído estão disponíveis sob a forma de bibliotecas de funções, podendo ser reutilizadas durante o desenvolvimento de novas aplicações informáticas distribuídas. O DCE é considerado um sistema flexível porque a sua implementação é independente tanto da rede informática como do sistema operativo onde reside. É também usado na construção e integração de aplicações cliente/servidor e fornece os serviços que simplificam o uso ou a chamada de serviços que podem estar fisicamente localizados noutro computador. Tende a ocultar a complexidade inerente ao processamento dos sistemas distribuídos tornando transparente e de fácil compreensão a construção, execução e manutenção de sistemas distribuídos. [Lockhart 1994]

O DCE fornece uma infra-estrutura completa do *Distributed Computing Environment*. Fornece serviços de segurança para proteger e controlar o acesso aos serviços de dados, a informação para que seja fácil encontrar os recursos distribuídos, e um modelo altamente flexível para organizar utilizadores dispersos, serviços, e dados. O DCE funciona em todas as plataformas principais e é projectado para suportar aplicações distribuídas em ambientes heterogéneos de hardware e de software. O DCE é uma tecnologia chave em três de áreas chaves: segurança, a *World Wide Web*, e objectos distribuídos. [The Open Group 2005]

2.1.1.1 Evolução Histórica

No ano de 1990, o DCE v1.0 começou a ser desenvolvido pela OSF (*Open Systems Foundatio*n), consórcio formado por mais de 200 empresas donde se destacam a IBM, Hewlett-Packard, Entegrity, Penn State e Chalmers. No ano seguinte saiu a primeira versão do DCE v1.0. Em 1994 foi disponibilizado o DCE v1.1 sob a licença da OSF resultando no *FreeDCE*. Em Março de 1996 foi apresentado o DCE1.2 e passados quatro anos realizou-se a primeira revisão, resultando daí o DCE 1.2.1. No dia 12 de Janeiro de 2005 foi lançada a versão DCE1.2.2 do DCE, pelo grupo "*The Open Group*", apresentando uma licença de software gratuito. [DCE 1996, DCE 2001, DCE 2005a]

2.1.1.2 Arquitectura

A arquitectura do DCE prevê a existência de um conjunto alargado de serviços organizados em vários componentes independentes que interagem entre si, tornando-o numa ferramenta

poderosa. [DCE 2005b] Os seus componentes podem ser vistos na Figura 4 **– Arquitectura do DCE [DCE 2005b]**.

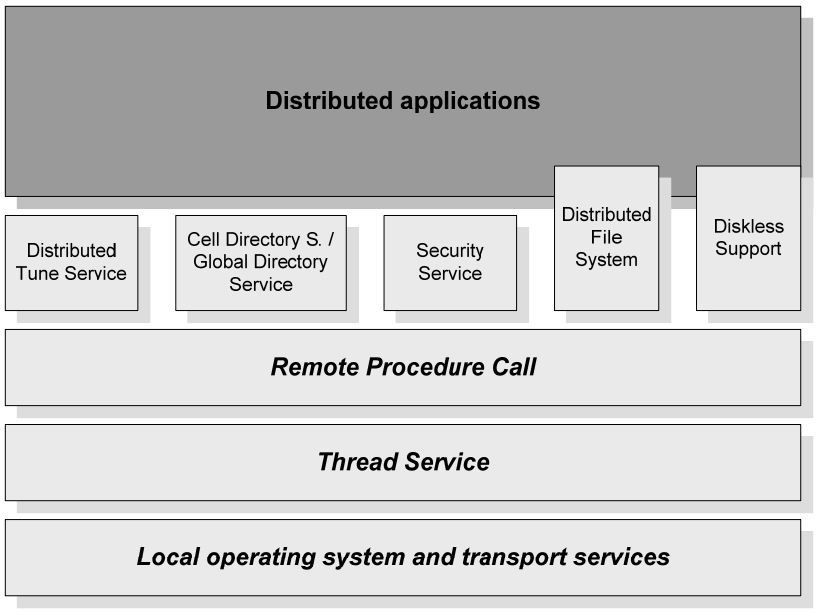

Figura 4 – Arquitectura do DCE [DCE 2005b]

O DCE engloba vários serviços, sendo eles os seguintes:

- Serviço de *Remote Procedure Call* chamado DCE/RPC;
- Serviço de Directórios (*DCE Directory Service*);
- Serviço de Segurança (*DCE Security System,* baseado no Kerberos);
- Serviço de Tempo Distribuído (*DCE Distributed Time Service*);
- Sistema de Ficheiros Distribuídos (*DTS – Distributed File System*).

Passa-se a explicar cada um dos serviços mais em pormenor.

DCE/RPC – este serviço é o componente principal do DCE, porque torna o desenvolvimento das aplicações distribuídas mais rápido e simples, fazendo uso de dois processos distribuídos, sendo eles a subrotina e o método de invocar. O primeiro encontra-se geralmente implementado no servidor, enquanto o segundo é usado na aplicação cliente. Os parâmetros inerentes à subrotina representam os dados que se pretendem passar quando o cliente questiona o servidor ou durante a resposta do servidor ao cliente.

O DCE/RPC fornece ao programador três serviços básicos:

- O meio de comunicação entre um cliente e um servidor.
- O mecanismo que permite ao cliente localizar o servidor numa rede.
- A transmissão dos dados na rede, convertendo-os nos formatos que achar conveniente.

Serviço de Directórios DCE – este serviço, idealizado para ser utilizado em sistemas distribuídos, fornece um mecanismo fiável onde as aplicações associam informações a nomes. O principal objectivo deste serviço é o de permitir aos clientes localizarem os servidores. No entanto, este serviço pode ser usado pelas aplicações que necessitam de disponibilizar o seu nome e os seus atributos na rede (aplicações que associam os seus nomes aos respectivos endereços, números de telefone, etc.).
O Serviço de Directórios DCE é composto por dois componentes:

- GDS (*Global Directory Service*);
- CDS (*Cell Directory Service*).

O CDS fornece a identificação do nó onde se inicia uma ramificação terminal chamada *Cell*. O GDS fornece a identificação da ramificação principal onde constam todos os CDS, isto é, todos os CDS aparecem como ramificações do GDS. O GDS está de acordo com a norma X.500 [X500 2006] no que respeita à identificação de serviços, ou seja, todos os sistemas que utilizarem o DCE como base podem ligar-se a outros sistemas exteriores, públicos ou privados, para aceder a serviços sem necessidade de recorrerem a alterações de software.
A *Web* usa o protocolo de identificação DNS (*Domain Name Service*). No DCE, o CDS pode usar o DNS como uma directoria global, substituindo-o ao GDS, permitindo aos utilizadores do DCE descobrir e interagir com outras redes existentes na *Web*. Cada utilizador apenas vê uma única árvore com várias ramificações onde constam os nomes ou *namespace* das aplicações disponibilizadas nas várias redes. Sendo um *namespace* distribuído, o Serviço de Directórios DCE consiste numa Base de Dados com um grande número de registos, independentes e localizados nos vários computadores que integram os vários nós. Assim, se houver um problema num dos nós do Serviço de Directórios DCE (computador existente na rede), não há impedimento que os restantes utilizadores acedam a outros computadores. O Serviço de Directórios DCE comporta-se como um sistema replicado, isto é, existem nele múltiplas cópias da identificação de um *namespace* nos vários nós do sistema. A replicação, por um lado ajuda a tornar o sistema mais fiável, já que elimina a possibilidade de falhas num ponto específico da rede e, por outro lado, permite aumentar a performance, visto a informação do *namespace* poder ser lida num nó mais próximo, reduzindo o tráfego de informação na rede. Para além destas vantagens, a replicação implica uma constante actualização das bases de dados, sendo ainda assim benéfica já que o número de leituras às bases de dados é largamente superior ao número de actualizações necessárias.

Serviço de Segurança DCE – este serviço fornece os mecanismos necessários para assegurar que os serviços existentes na rede estejam apenas disponíveis para utilizadores autorizados. Este problema não se coloca caso os utilizadores se encontrem num computador local, dado que os sistemas de segurança são assegurados localmente. O problema de segurança coloca-se somente nos sistemas ou redes distribuídas e pode ser descrito em três partes:

Ao estabelecer uma ligação com um determinado utilizador, algures numa localização da rede, tem que se ter a certeza que essa pessoa é a certa (Autenticação).

É necessário determinar quais os privilégios associados a cada utilizador (Autorização).

É necessário verificar se esses privilégios permitem a realização das operações desejadas pelos utilizadores, entre elas a que controla a Base de Dados da segurança onde estão as encriptação das chaves e password dos utilizadores (Serviço de registos).

Em sistemas distribuídos é imperativo que cada utilizadores faça o seu registo na rede e não num computador específico, porque assim garante-se que um utilizador não autorizado não consiga entrar na rede, tenha acesso às informações que lá circulam e consiga ler informações confidenciais, palavras-chave ou até mesmo, enviar mensagens em nome de um utilizador autorizado para essa rede. Portanto, um sistema distribuído para ser robusto, seguro e fiável não deve permitir o acesso a serviços se o sistema de segurança não estiver activo. O Sistema de Segurança DCE vem resolver todos estes problemas. Tal como o Serviço de Directórios DCE, o Sistema de Segurança DCE é também distribuído e replicado, autentica os utilizadores, determina as suas autorizações, é imune aos ataques que envolvem a captura de mensagens na rede e nunca transmite palavras-chave.

Serviço Tempo Distribuído DCE – este serviço é responsável pela sincronização dos relógios dos vários computadores existentes na rede. Se num dos nós da rede existir um computador remoto oficial que forneça as horas GMT (*Greenwich Mean Time***)**, o Serviço Tempo Distribuído DCE actualiza os relógios de todos os computadores com a hora fornecida por esse computador oficial. Esta sincronização dos relógios torna-se um aspecto importante quando o protocolo utilizado pelo Serviço de Segurança assume que todos os nós da rede têm a mesma hora. Existem sistemas de segurança que fornecem uma licença cuja duração caduca com o tempo de utilização. Existem ainda programas do tipo “cópias de segurança” que são controlados e lançados pela data de criação dos ficheiros a salvaguardar. Se houver uma discrepância nos relógios dos vários computadores, pode dar-se o caso destas licenças expirarem prematuramente ou de o processo criação de cópias de segurança não ser realizado correctamente.

Sistema Distribuído de Ficheiros – este sistema fornece os mecanismos para as aplicações clientes poderem aceder a ficheiros localizados no servidor, como se estivessem localizados localmente. Todos os nós na rede identificam o mesmo ficheiro pelo mesmo nome e vêem-no localizado no mesmo endereço. O Sistema Distribuído de Ficheiros também pode utilizar a replicação de dados nos diferentes nós, mantendo desta forma a consistência entre as cópias. O número e a localização das cópias são controlados pelo administrador do sistema. Este sistema é implementado no topo da pirâmide dos serviços DCE, como se pode ver na Figura 4, podendo ser dito que o Sistema Distribuído de Ficheiros possui toda a interoperacionalidade do DCE.

2.1.2 CORBA

O CORBA (*Common Object Request* ***Broker*** *Architecture*) foi desenvolvido pelo OMG (*Object Management Grou*p). [OMG 2006, OMG 2006b] Esta organização congrega mais de 800 empresas e tem como objectivo facilitar o desenvolvimento de aplicações distribuídas. Estão disponíveis no mercado implementações desta arquitectura, sob a forma de bibliotecas de funções que permitem aos programadores desenvolver mais facilmente e rapidamente aplicações distribuídas.

O CORBA constitui um mecanismo fiável que permite estabelecer e simplificar a troca de dados entre sistemas distribuídos e heterogéneos. Face à diversidade de hardware e software utilizados pelos programadores no desenvolvimento de sistemas distribuídos, o CORBA é o responsável pela comunicação e interacção destas aplicações com outras existentes dentro da mesma rede, tornando-as totalmente transparentes. Assim, pretende-se que uma aplicação distribuída consiga interagir com outras aplicações, independentemente da linguagem de programação, da sua localização, dos seus serviços e dos sistemas operativos. No CORBA, mais importante do que o método ou a forma como as aplicações são implementadas, é fundamental o modo como os objectos são referenciados e como são acedidos no universo dos restantes objectos existentes na rede. Assim, esta tecnologia disponibiliza transparência face à localização dos objectos, às linguagens e às redes, permitindo integrar aplicações já existentes e reduzir custos de manutenção.

O CORBA é um dos modelos mais populares de objectos distribuídos juntamente com o DCOM (*Distributed Component Object Model*) da MICROSOFT.

2.1.2.1 História e Evolução

Em Outubro de 1991, o CORBA 1.0 foi apresentado como versão inicial que incluía: o modelo de objectos CORBA, a linguagem da definição de relação IDL (*Interface Definition Language*) e as relações de programação de aplicações API (*Aplication Programming Interfaces*) para pedidos de gestão e invocação dinâmicos DII, integrando ainda uma biblioteca para a linguagem de programação C. Em Fevereiro de 1992 foi publicado o CORBA 1.2, a primeira versão da especificação do CORBA, que permitiu corrigir algumas anomalias da especificação original, acrescentar definições de relação e optimizar a gestão de memória em relação ao modelo original. Em Agosto de 1996, o CORBA 2.0 sofreu a primeira revisão geral, tendo-lhe sido acrescentadas novas funcionalidades. Em Junho de 1999, a versão CORBA 2.3 sofreu novas revisões e foram-lhe adicionadas outras especificações (Portabilidade IDL/Java, Biblioteca de Java para IDL e de IDL para Java, Biblioteca de C++, entre outras). Em Outubro de 2000 foram incluídas novas especificações, surgindo o CORBA 2.4 (CORBA em tempo real, Núcleo e RTF2.4). Em Setembro

de 2001 deu-se uma nova revisão surgindo o CORBA 2.5, que incluiu algumas novas especificações (optimização da transmissão de mensagens, CORBA Tempo Real). Em Julho de 2002, o CORBA 3.0 fez o CORBA Mínimo e CORBA em Tempo Real aparecerem como documentos separados (ambos tinham sido acrescentados ao Núcleo CORBA no Lançamento 2.4).

2.1.2.2 Arquitectura

A arquitectura CORBA permite a um **Cliente** aceder a recursos computacionais e informações distribuídas a partir de aplicações correntes. Tem como objectivo definir e proporcionar interfaces de programação para os chamados componentes ORB (*Object Request **Broker***). O ORB é um mecanismo básico que permite que aplicações distribuídas comuniquem entre si de uma forma transparente, quer se encontrem localmente na mesma máquina, quer se encontrem algures num computador existente numa rede.

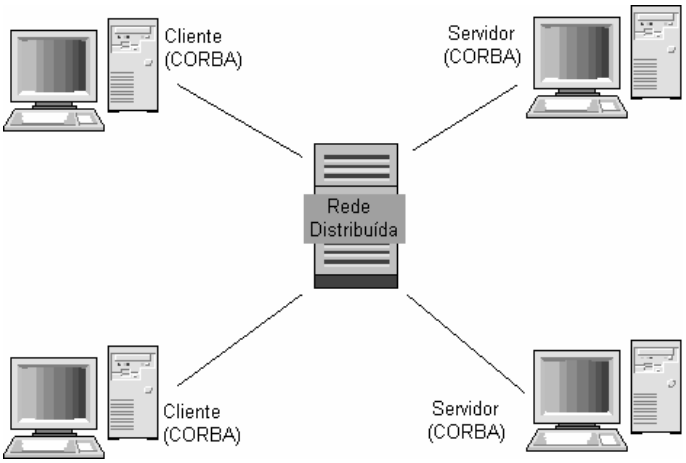

Figura 5 – Rede computacional distribuída (CORBA).

Um **Cliente** não necessita saber quais os mecanismos para activar ou comunicar com uma determinada aplicação, nem como esta é implementada, nem mesmo qual a sua localização. O ORB constitui uma base para construir aplicações distribuídas e proporcionar uma interoperacionalidade entre aplicações tanto em ambientes homogéneos como heterogéneos.
O conceito de objecto local (**Cliente**) e objecto remoto (servidor) também são utilizados na arquitectura CORBA para definir os objectos envolvidos na chamada de um método remoto. Na Figura 6 pode-se ver como é feito um pedido de um **Cliente** a um servidor localizado num sistema distribuído. [Santos 1996] No final deste processo, o **Cliente** tem que saber localizar o servidor, interagir com ele e conseguir chegar ao *Object Implementation* via ORB. No entanto, a aplicação **Cliente** tem primeiramente que saber o nome do serviço que pretende invocar para o transmitir ao ORB. Este vai pesquisar, numa espécie de Base de Dados chamada IR (*Interface Repository*),

onde constam os nomes dos serviços ou objectos e as suas localizações na rede, identificando, de forma unívoca, o objecto solicitado pelo **Cliente**. Todo o objecto tem que escrever a sua identificação e localização no IR, visto ser esta a única forma de poder ser identificado e invocado por aplicações distribuídas. O *Client Interface,* objecto do lado do **Cliente**, é definido recorrendo à IDL *(Interface Definition Languag*e), que é uma linguagem genérica para descrever interfaces e não uma linguagem de programação. A sintaxe da IDL apenas permite definir os métodos e os seus valores de retorno, não estando nela contemplados os fluxos de controlo. [Vinoski 1997]

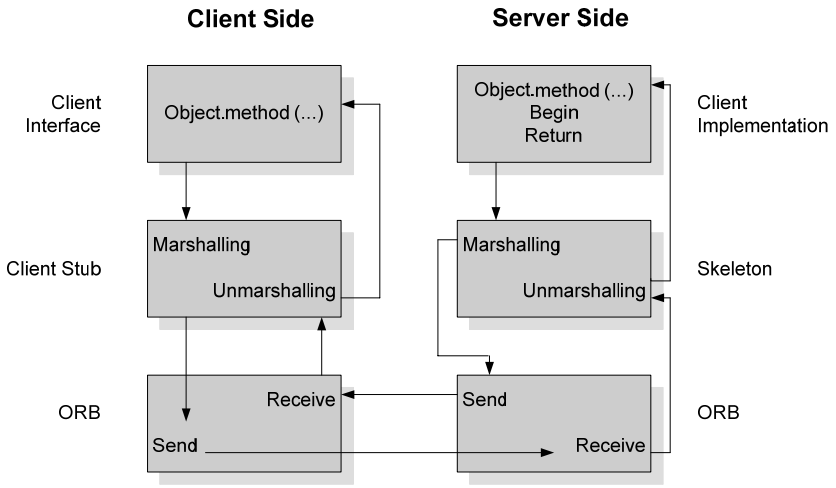

Figura 6 – Pedido de um objecto de um *Cliente* CORBA.

***ORB* –** fornece um mecanismo que permite a um **Cliente** comunicar, de uma forma transparente, com aplicações que se encontram localizadas num sistema distribuído. Também simplifica a programação, a localização das aplicações distribuídas, sendo ainda responsável pela transmissão e recepção das mensagens previamente formatadas pelo componente *Stub*.

***Client Stub* –** é neste componente do CORBA que ocorre, de uma forma automática e sem intervenção do programador, a formatação dos parâmetros a serem transmitidos pelo ORB para a rede (*marshlling*). O ORB ao receber uma mensagem do servidor vai fazer o inverso (*unmarshlling*), reconfigurando a mensagem recebida da rede para uma mensagem capaz de ser compreendida pelo **Cliente**.

***Skeleton* –** é este componente do CORBA, localizado no lado do servidor, que vai chamar o objecto pretendido e executar a sua implementação quando recebe um pedido do **Cliente** via ORB. Realiza da mesma forma que o *Client Stub* as operações de *marshling* e *unmarshling* às mensagens transmitidas e recebidas da rede.

2.1.3 Web Services

Os negócios têm vindo, cada vez mais, a transformarem-se de processos de negócio baseados em transacções para uma aproximação orientada a serviços, muitas vezes invocando a colaboração de múltiplas organizações para fornecer um serviço integrado aos **Clientes**. As organizações não podem continuar a existir no isolamento. Para competir, de uma forma eficiente, têm de formar alianças estratégicas para fornecer serviços aos seus **Clientes**.

As cadeias de fornecimento tradicionais têm sido substituídas por parcerias de serviços, que colaboram nos bastidores para fornecer serviços integrados aos consumidores. Isto significa que, não só o relacionamento entre organizações muda, como os sistemas de informação também terão de evoluir se quiserem suportar o novo paradigma de negócio. Adicionalmente, a curva descendente da economia actual significa uma pressão, cada vez maior, para consolidar sistemas de informação e capitalizar investimentos existentes. Consequentemente, é essencial uma forma efectiva de reutilizar componentes e integrar serviços.

Os métodos existentes de reutilização e integração de aplicações requerem desenvolvimentos dispendiosos e morosos. E mesmo que tenhamos sucesso a integrar as aplicações, fazer alterações a alguma delas ou acrescentar uma nova pode quebrar essa integração. O desafio para a indústria de tecnologias de informação tem sido o desenvolvimento de uma plataforma "*lightweight*", que possa ser usada para satisfazer a reutilização de requisitos modulares de uma forma eficiente em termos financeiros. A tecnologia dos *Web Services* oferece uma forma de atingir estes objectivos.

Os *Web Services* tornaram-se uma tecnologia emergente em 2002, depois de terem sido apresentados pela MICROSOFT em 2000, como um dos maiores componentes da sua tecnologia .NET, com o objectivo de revolucionar a computação distribuída.

Os *Web Services*, numa definição mais abrangente do conceito, são serviços disponibilizados a outras aplicações, através da *World Wide Web*, ou de uma intranet, e usam um sistema de troca de mensagens baseado em XML standard. Podem operar sobre qualquer rede (a Internet ou redes privadas intranet) para executar determinadas tarefas. Os pedidos podem ser enviados e as respostas recebidas entre duas diferentes aplicações em dois computadores separados pertencentes a diferentes unidades de negócio. Os **Clientes** destes serviços podem agregá-los, de modo a formar uma aplicação *end-user*, ou a criar novos *Web Services*.

Por outras palavras, toda a aplicação que possa ser acedida pela Internet usando algum protocolo de comunicação (por exemplo, HTTP, SMTP), e algum tipo de codificação (por exemplo, XML) é

um *Web Services*. Existem, provavelmente, tantas definições de *Web Services* como empresas a desenvolvê-los, mas quase todas estas definições têm algo em comum[1].

Os principais participantes do modelo de arquitectura orientada aos serviços são: o **Cliente**, o **Fornecedor** e o servidor UDDI. [Ribeiro 2005]

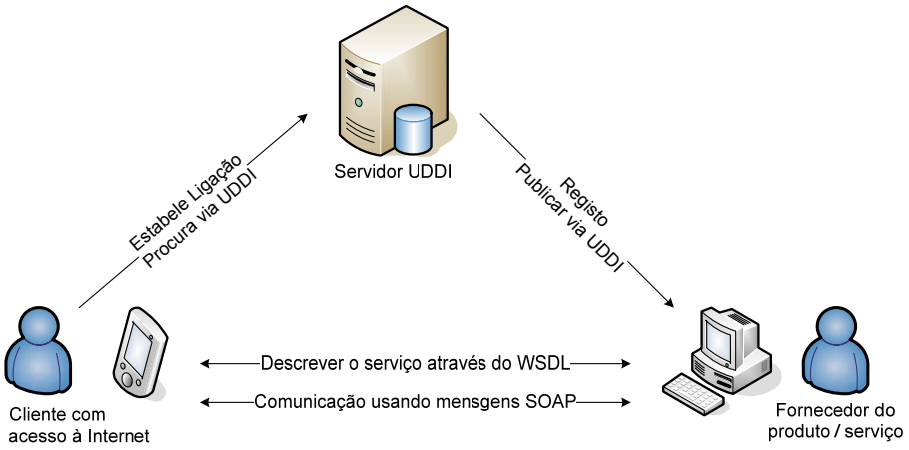

Figura 7 – Modelo genérico de um *Web* Service.

Para além de utilizarem as especificações padrão da *Web* (SOAP, WSDL, UDDI, HTTP), estas aplicações podem recorrer a outras plataformas de codificação, transporte, registo e/ou pesquisa de serviços *Web.*

Os *Web Services* são uma tecnologia emergente, sobre a qual muito se tem escrito. Enquanto que uns a consideram como um caminho a seguir no âmbito do desenvolvimento de aplicações distribuídas, outros apenas vêem nela mais uma evolução de um conceito antigo (CORBA).

Existem várias definições para os *Web Services*, passo a indicar algumas:

Segundo W3C Serviços *Web* Arch WG: "Um *Web* Service é uma aplicação de software identificada por uma URI (Uniform Resource Identifier) cujas interfaces e ligações são capazes de serem definidas, descritas e descobertas por artefactos XML. Suportam interacções directas com outras aplicações de software usando mensagens baseadas e, XML via protocolos baseados na Internet". [W3C 2006]

Outra definição segundo a W3C: "*A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically* WSDL*). Other systems interact with the Web service in a manner prescribed by its description using* SOAP *messages, typically conveyed using* HTTP *with an* XML *serialization in conjunction with other* [*Web*-related standards.]". [W3C 2004]

[1] www.sinfic.pt

Segundo Java Technology: "*Web Services* são aplicações empresariais baseadas na *Web* que usam o XML standard e protocolos de comunicação para trocar dados entre si". [Java 2006]
Segundo a Gartner Group " […] componentes de software espalhados que interagem dinamicamente uns com os outros através de tecnologias Internet standard[…]"[Gartner 2006]
Segundo a Forrester Research " […] ligações automáticas entre pessoas, sistemas e aplicações que expõem elementos de funcionalidade de negócio como um serviço de software e criam um novo valor de negócio […] " [Forrester 2006]
Uma outra definição diz que um "*Web Service* é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Os *Web Services* são componentes que permitem às aplicações enviar e receber dados em formato XML. Cada aplicação pode ter a sua própria "linguagem", que é traduzida para uma linguagem universal, o formato XML.
Para as empresas, os *Web Services* podem trazer agilidade para os processos e eficiência na comunicação entre cadeias de produção ou de logística. Toda e qualquer comunicação entre sistemas passa a ser dinâmica e principalmente segura, pois não há intervenção humana."[Wikipédia 2007]
Segundo outro autor, Márcio Boaro, "Um *Web* Service é uma aplicação lógica, programável, acessível, que usa os protocolos padrão da internet, para que se torne possível a comunicação transparente de máquina para máquina e aplicação para aplicação. Para desenvolvermos um *Web* Service utilizamos tecnologias pensadas sob esta visão, como o SOAP, *WSDL* HTTP, usadas para passar mensagens através das máquinas. Estas mensagens podem variar muito na complexidade, desde a chamada de um método, às submissões da ordem de compra". [Boaro, Marcio 2006]

Figura 8 – Arquitectura em Camadas *Web Service.*

O ciclo de vida de um *Web Service* (Figura 9) pode dizer-se que tem quatro fase distintas, sendo elas as seguintes:

- Fase do Registo;
- Pesquisa;
- Descrição;
- Invocação.

Estes estados podem ser utilizados directamente ou indirectamente pelas aplicações, ou seja, directamente se se aceder a página do **Fornecedor** e fizer um registo ou pesquisa e indirectamente se for a aplicação do **Cliente** a fazer essas operações.

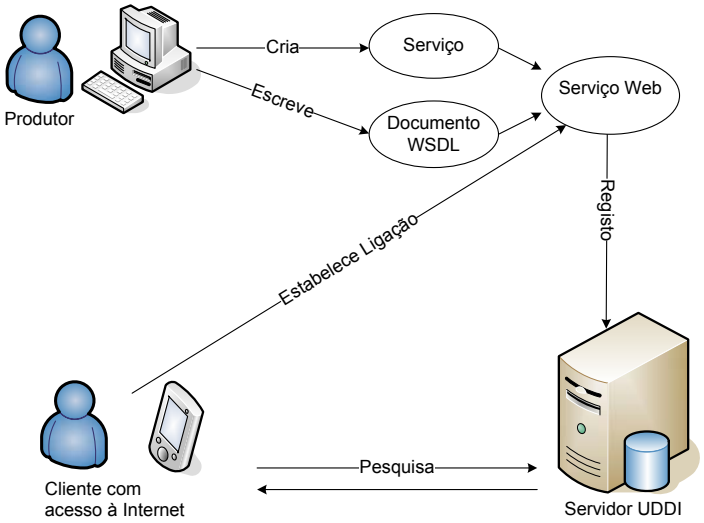

Figura 9 – Ciclo de vida de um *Web Service.*

O ciclo de vida de um *Web Service* contempla quatro etapas principais [Lopes 2004]:

- A primeira etapa consiste na construção e registo do *Web Service*. Este pode ser programado em qualquer linguagem de programação que permita interpretar XML, em seguida o **Fornecedor** gera um documento WSDL para descrever os *Web Services* suportados pelo seu processador SOAP (Descrição).
- A segunda etapa consiste no registo do *Web Service* num dos servidores UDDI. Nesta fase, são fornecidas aos servidores UDDI, todas as informações respeitantes ao *Web Service* que se pretende disponibilizar, o **Fornecedor** usa as APIs(*Application Programming Interface*) do UDDI para registar esta informação junto servidor UDDI, Depois da submissão dos dados do *Web Service* e dos demais dados de contacto, a entrada no registo conterá uma URL que apontará para o site onde estiver o servidor SOAP com o arquivo WSDL ou outro esquema XML que descreverá o *Web Service*. (Registo).

- A terceira etapa consiste na pesquisa de um *Web* Service, o **Cliente** (processador SOAP) consulta o registo nos servidores UDDI por parte de um utilizador (aplicação **Cliente**). Nesta etapa deverá ficar a conhecer-se a localização do *Web* Service pretendido, os seus parâmetros de entrada e saída, o nome dos serviços ou as funções disponibilizadas, etc. (Pesquisa).
- Na quarta etapa dá-se o estabelecimento da ligação ou comunicação entre o utilizador (aplicação **Cliente**) e o **Fornecedor** *Web Service* através de mensagens SOAP/XLM. É claro que, tanto o **Cliente** como o servidor devem ser capazes de operar com o mesmo protocolo (neste caso, SOAP sobre HTTP) e compartilhar a definição de serviços (que está representada com WSDL).

Passa-se a demonstrar comparação entre o uso de um *Web Service*, incluindo o *Web Service do Google,* para o levantamento de dinheiro num Banco comparando como o que se faz actualmente. O processo está descrito em 7 passos, são eles os seguintes:

- O **Cliente** descobre o *Web Service*. Durante o acto da descoberta, por exemplo, o **Cliente** pode fazer o download de um ficheiro que explique como interagir com o *Web Service*. Esta etapa corresponde a fase de quando alguém se dirigi ao Banco para pedir informações. O Banco indica as regras para fazer se um levantamento ou o funcionário pode indicar ao **Cliente** quais são as regras internas desse Banco.
- O **Cliente** faz um pedido de levantamento de dinheiro baseado nas regras divulgadas pelo *Web Service* durante a fase da descoberta. As regras podem especificar que o pedido tem que aparecer em um determinado formulário, e o **Cliente** deve fornecer dados específicos. Esta etapa é a mesma que a pessoa que anda até à caixa com um pedido de levantamento de dinheiro. O pedido deve conter o número de **Cliente** da pessoa, a quantidade que desejam se levantar, e a outra informação necessária. O Banco especifica o formato do pedido e da informação que deve conter.
- O utilizador pode pedir ao **Cliente** credenciais dependendo do *Web Service*. Os *Web Service* do Google são públicos mas requerem ainda que se forneça um número de licença do colaborador (**Cliente**) como sua identificação. Esta etapa é a mesma que o funcionário do Banco lhe pede outro formulário da licença ou da identificação antes de formalizar o pedido de levantamento.
- O *Web Service* executa o trabalho requerido para formalizar o pedido. A maioria de argumentos, os acessos do *Web Service* são uma Base de Dados para a informação, poderia incorporar uma ordem, e pode mesmo fornecer algum nível da informação do formato sobre a informação original. Os *Web Service* do Google executam um número de tarefas dependendo do pedido que se faça. O pedido mais fácil é uma busca geral, mas você pode também executar verificações tais como fazer uma verificação de ortografia. Esta etapa iguala à caixa de Banco que começa contar o dinheiro.

- O *Web Service* emite os dados ao **Cliente**. O índice da informação depende do *Web Service*. Os *Web Service* de Google fornecem dados num formato muito específico baseado no índice da Base de Dados associada e da natureza do pedido. Esta etapa corresponde à fase em que o funcionário entrega o dinheiro à pessoa. No general, o funcionário conta o dinheiro em frente à pessoa, melhor que é melhor que apenas entregar o dinheiro.
- O **Cliente** faz *logout Web Service* ou o *Web Service* desconecta o **Cliente** após um período de inactividade. Esta etapa corresponde à pessoa que sai do Banco com o seu dinheiro. Se a pessoa não sair do Banco, acabaram por pedir para sair, se não for mais na hora de fecho.
- O **Cliente** faz algo com os dados que recebe. Em muitos casos, formata os dados e os mostra no ecrã do utilizador. Esta etapa corresponde à fase em que se gasta o dinheiro que se recebeu do Banco.

Veja-se mais pormenorizadamente cada um dos estados dos Serviços *Web*:

- **Descrição –** processo pelo qual o *Web Service* expõe o seu API mais especificamente o seu documento WSDL (*Web Services Description Language*), tendo a aplicação **Cliente** acesso a toda a interface do *Web Service*. Desta forma, a aplicação **Cliente** passa a dispor as informações de todas as funcionalidades disponibilizadas pelo *Web Service*, assim como os métodos, protocolos e tipos de mensagens que permitem aceder às suas funcionalidades. Neste estado é utilizada a norma WSDL.
- **Registo –** processo opcional através do qual o **Fornecedor** do *Web Service* dá a conhecer a sua existência e a do seu serviço através de um registo num dos servidores UDDI existentes (IBM, MICROSOFT, GOOGLE). Neste estado é utilizada a norma UDDI.
- **Pesquisa –** processo opcional através do qual uma aplicação **Cliente** toma conhecimento da existência do *Web Service* pretendido, através de uma pesquisa nos servidores UDDI. Neste estado é utilizada a norma UDDI.
- **Invocação –** processo pelo qual o **Cliente** e o servidor interagem através do envio de mensagens de ***"input"*** e de eventuais recepções de mensagens de ***"output"***. Neste estado é utilizada a norma SOAP ou XML.

2.1.3.1 SOAP

SOAP é um protocolo projectado para invocar aplicações remotas através de RPC (*Remote Procedure Calls* - Chamadas Remotas de Procedimento) ou trocas de mensagens, em um ambiente independente de plataforma e linguagem de programação. SOAP é, portanto, um padrão normalmente aceito para utilizar- se com *Web Services*. Desta forma, pretendesse garantir a

interoperabilidade e intercomunicação entre diferentes sistemas, através da utilização de uma linguagem (XML) e mecanismo de transporte (HTTP) padrões.

2.1.3.1.1 História e Evolução

No Verão de 1998, a USERLAND SOFTWARE (Dave Winer, Don Box), uma empresa aliada da MICROSOFT, apresentou uma especificação chamada XML-RPC. No ano de 1999, a MICROSOFT lançou a primeira versão do SOAP1.0. Em Maio de 2000, Don Box, DEVELOPMENTOR, IBM e a MICROSOFT apresentaram a versão SOAP1.1 numa nota do W3C. Em Dezembro de 2001 foram publicados os primeiros artigos sobre SOAP1.2. Em 2004, a W3C apresentou a última versão do SOAP (SOAP1.2) como uma especificação. [SOAP 2005a, SOAP 2005b]

2.1.3.1.2 Estrutura das Mensagens SOAP

As mensagens SOAP são constituídas pelas seguintes 3 estruturas:

- **Envelope (*'Envelope'*) –** Toda mensagem SOAP deve contê-lo. É o elemento raiz do documento XML. O Envelope pode conter declarações de *namespace* (tem como *namespace* obrigatório o endereço HTTP://schemas.XMLSOAP.org/SOAP/envelope)e também atributos adicionais como o que define o estilo de codificação, ou seja, orma como os dados são representados na mensagem(*encodingStyle*). Um *encodingStyle* define como os dados são representados no documento XML. O início das mensagens SOAP identifica-se através da etiqueta *"<envelope"* e termina com a etiqueta *"/envelope>".*
- **Cabeçalho (*'Header'*) –** Este elemento encontra-se dentro do elemento *Envelope* e começa com a etiqueta *"<header",* terminando com a etiqueta *"/header>"*. É um cabeçalho opcional. Ele carrega informações adicionais, como por exemplo, se a mensagem deve ser processada por um determinado nó intermediário (É importante lembrar que, ao trafegar pela rede, a mensagem normalmente passa por diversos pontos intermediários, até alcançar o destino final). Quando utilizado, o Header deve ser o primeiro elemento do Envelope.
- **Corpo ('Body') –** Este elemento é obrigatório e contém o payload, ou a informação a ser transportada para o seu destino final. O elemento Body pode conter um elemento opcional Fault, usado para carregar mensagens de status e erros retornadas pelos "nós" ao processarem a mensagem. Insere-se igualmente dentro do elemento *Envelope* e o seu início pode ser identificado através da etiqueta *"<body"* e o seu final através da etiqueta *"/body>"*.
- **Anexo ("Attachment")** – este elemento é opcional e encontra-se definido dentro do elemento envelope. Podem existir vários elementos deste tipo num <envelope>. É

normalmente usado para transmitir anexos, recorrendo a mecanismos MIME (Multipurpose Internet Mail Extensions).

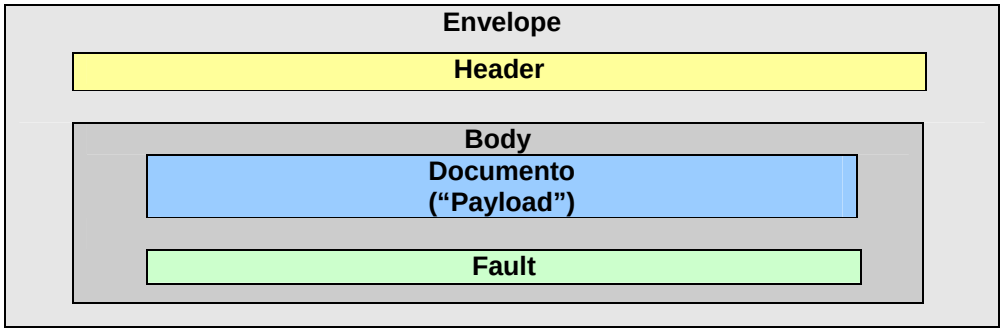

Figura 10 – Estrutura da mensagem SOAP.

De forma a perceber a estrutura de uma mensagem SOAP é normal estabelecer a analogia com uma carta de correio. [Lopes 2005] A razão desta analogia reside no facto da estrutura destas mensagens estar dividida em quatro secções.

Em seguida apresenta-se um exemplo de uma mensagem SOAP:

```
<SOAP-ENV:Envelope
        XMLns:SOAP-ENV = "HTTP://schemas.XMLSOAP.org/SOAP/envelope/"
        SOAP-ENV:encodingStyle="HTTP://schemas.XMLSOAP.org/SOAP/encoding/">
        <SOAP-ENV:Header>
                <t:Transaction
                XMLns:t="some-URI" SOAP-ENV:mustUnderstand="1">
                <t:Transaction>

        </SOAP-ENV:Header>
        <SOAP-ENV:Body>
                <m:GetLastTradePrice> XMLns:m = "Some-URI">
                <symbol>DIS</symbol>
                </m:GetLastTradePrice>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

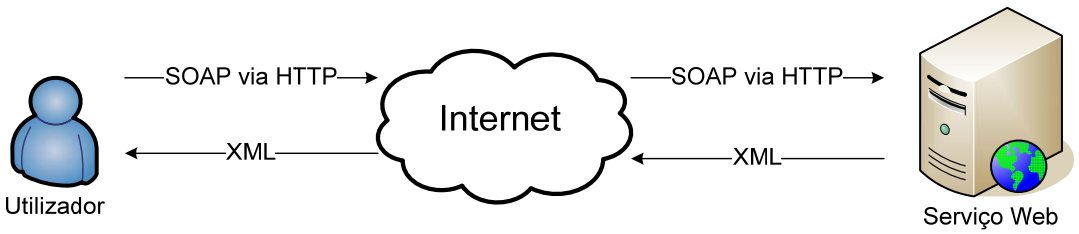

Figura 11 – SOAP / XML.

2.1.3.2 WSDL- *Web Services Description Language*

A descrição do *Web* service é feita através de um arquivo WSDL, cujo formato é XML. Este arquivo tem como objectivo descrever a interface do serviço, ou seja, os métodos que são disponibilizados pelo *Web Service*, e também os parâmetros recebidos, a resposta enviada e ainda o processo de comunicação com o servidor SOAP, a forma de o invocar, as operações disponibilizadas pelo **Fornecedor**, o protocolo de comunicação e o tipo de dados utilizados.
É através de WSDL que o UDDI descreve os detalhes de localização e chamada ao *Web Service*.
Um **Cliente** SOAP vai ler a estrutura do arquivo WSDL e, a partir dos dados ali referenciados, comunicar-se com o servidor SOAP para acesso ao serviço descrito.
A geração do arquivo WSDL pode ser feita de forma manual ou automática, através de ferramentas tais como WSTK – *Web* Service Tool Kit da IBM ou o utilitário WSDL.EXE na plataforma .NET.
Há 4 tipos de operações que se podem realizar com o WSDL:

- Pedido – o servidor recebe um pedido, mas não envia resposta.
  Ex. Submissão de dados num site de venda de livros.
- Pedido/Resposta – o servidor recebe um pedido e processa a resposta
  Ex. Motor busca SAPO.
- Solicitação/Resposta – o servidor envia uma mensagem ao **Cliente** e recebe uma mensagem de resposta. É o oposto do Pedido/Resposta, visto o **Cliente** ser o servidor.
  Ex. Pedido de identificação por parte de um servidor.
- Operação de Notificação – é uma mensagem enviada pelo servidor, e para o qual não é esperada resposta.
  Ex. Envio de uma mensagem para recuperação de password.

2.1.3.2.1 História e Evolução.

Em Março de 2001, o WSDL 1.1 foi sugerido numa nota do W3C (pelas empresas ARIBA, IBM e MICROSOFT) como um documento utilizando a especificação XML para descrever Serviços *Web*. Esta nota também referia como usar o WSDL juntamente com SOAP1.1, HTTP GET/POST e MIME. Em Dezembro de 2001 foram publicados os primeiros artigos sobre o WSDL1.2. Em Julho de 2002 foi criado o primeiro serviço a utilizar o WSDL1.2 e em Agosto de 2005 foi implementada a primeira versão do WSDL2.0. [WSDL 2006a]

2.1.3.2.2 Estrutura dos Documentos WSDL

Um documento WSDL obedece a uma estrutura conhecida e bem definida, onde cada elemento tem uma função clara e objectiva. Esta estrutura, representada na Figura 12, assenta num elemento principal chamado *Definitions*, no qual podem existir cinco sub-elementos. [Menéndez 2002]

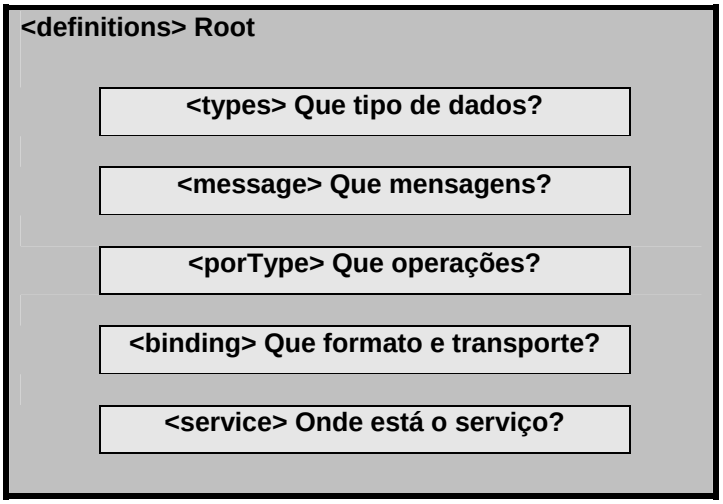

Figura 12 – Estrutura de um documento WSDL.

- ***definitions*** **–** elemento raiz que contém todas as definições respeitantes a todos os elementos presentes no ficheiro WSDL, o atributo *targetNamespace* permite definir um namespace(funciona como etiqueta que permite identificar univocamente os diversos nomes) e atributo XML*ns* permite referenciar outros namespace.

  <*definitions*>
  XMLNS:HTTP = HTTP://schemas.XMLSOAP.org/WSDL/HTTP/"
  (...)
  </*definitios*>

- ***types*** – elemento onde constam as informações que identificam o tipo de dados contidos nas mensagens. Para ter maior integração, os *Web Services* usam XML Shemas para definir os tipos de dados das mensagens (W3CXMLSCHEMA). Estrutura de dados do tipo abstracto, uma vez que são independentes do protocolo a utilizar

  <*types*>
  Definição dos tipos de dados...
  </*types*>

- ***messages*** **–** elemento que descreve de uma forma abstracta os fluxos de informação, nomeadamente os nomes dos parâmetros necessários para invocar correctamente cada uma das funções disponibilizadas pelo *Web Service,* representados pela notação *message parts*. Cada parâmetro tem um *types* previamente definido.

<*message name* = “NomeLivroSOAPin”>
<***part name***=“parameters”
element=“S0: Nome Livro” />
</*message*>
<*message name* = “NomeLivroSOAPout”>
<***part name***=“parameters”
element=“S0: Nome Livro Response” />
</*message*>

- ***portTypes*** **–** elemento mais importante do WSDL onde são descritas as operações (métodos) disponibilizadas no *Web Service*.Define o tipo de messagem envolvidas. São compostos por um conjunto de mensagens de *input* e *output,* podendo ainda ser encontradas mensagens de erro (*operation*).

  <*portType*>
  Definição do tipo de port...
  </*portType*>

- ***bindings*** **–** elemento que identifica o formato da mensagem e protocolo de transporte. Constituído por 2 elementos *name* e *type*:
  - ***name*** **–** define o nome, pode ser o que quisermos;
  - ***Type*** **–** Indica o protocolo de transporte utilizado.

    <***Binding name*** = “Preço_X0020 _ LivroSOAP”
    <*part name*=“parameters”
    type =“S0: Preço_X0020 _ LivroSOAP” >

- ***services*** **–** elemento constituído pelo conjunto de elementos *port* que identificam os diferentes pontos de acesso/localização do *Web Service* a invocar, normalmente através do seu endereço URL. Encontra-se ainda neste elemento a localização do endereço do *Web Service* (*Ports*).

- ***documentation*** **–** elemento destinado a comentários, podendo conter texto como elementos XML. Pode encontrar-se dentro do elemento *definitions* ou dentro de qualquer um dos outros elementos do documento WSDL.

  <*documentation*>
  Loja de Livros com os melhores preços do Mundo
  </*documentation*>

2.1.3.3 UDDI

O UDDI é como uma Base de Dados de informação sobre *Serviços Web*. É uma plataforma independente que permite disponibilizar, utilizar e pesquisar serviços na Internet. O UDDI é uma interface de serviço descrita através de documentos WSDL, que constitui uma parte integrante da plataforma .NET da Microsoft e que comunica utilizando mensagens SOAP transmitidas através do protocolo HTTP. Assim, o UDDI tem como objectivos proporcionar a um **Fornecedor** de serviços *Web* ou a um utilizador (**Cliente**), uma ferramenta simples, rápida e automática de:

- Encontrar o serviço pretendido entre os milhares que estão dispostos na rede;
- Definir como é realizado a transacção quando é descobrir o **Fornecedor** ideal;
- Definir como configurar a comunicação, uma vez que o serviço foi escolhido;
- Permitir descobrir **Clientes** novos e aumentando o acesso aos **Clientes** actuais;
- Permitir, de forma fácil, o acesso a novos serviços, assim como aos já existentes;
- Remover as barreiras para permitir a participação rápida na economia global do Internet;
- Descrevendo serviços e processos do negócio automaticamente num ambiente universal.

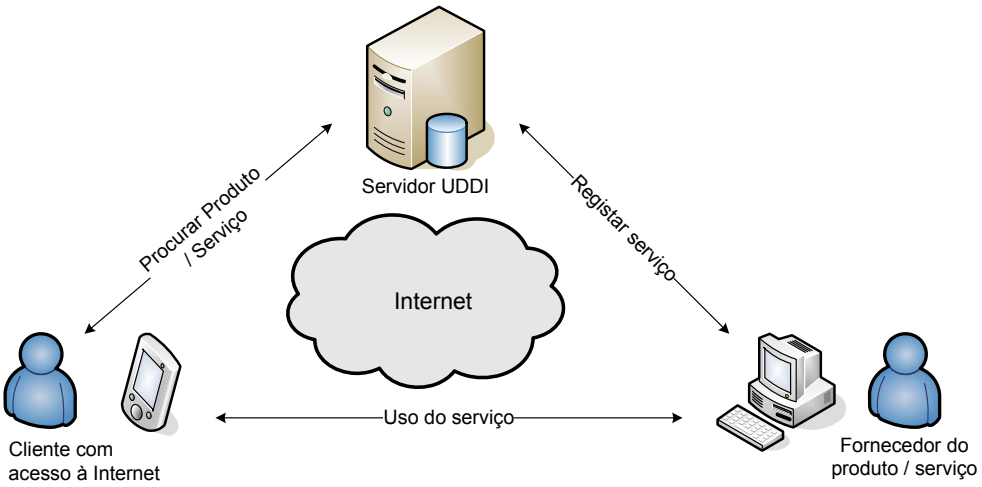

Figura 13 – Serviços UDDI.

A Figura 14 mostra a localização do UDDI numa pilha protocolar onde se encontram outras tecnologias e especificações já referidas nas secções anteriores.

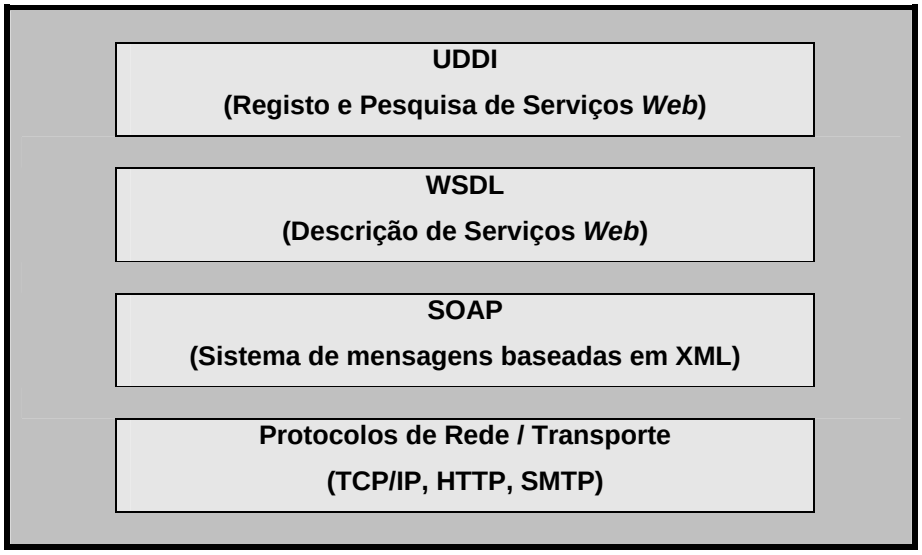

Figura 14 – Pilha protocolar do UDDI.

2.1.3.3.1 História e Evolução

Projecto iniciado pelas empresas: IBM, Microsoft e Ariba com o intuito de criar um método standard capaz de divulgar e descobrir *Web Services*,mas que ao mesmo tempo pudesse ser invocado independentemente da linguagem ou plataforma de desenvolvimento. Em Setembro de 2000, o UDDI v1.0 foi publicado de forma a permitir o registo de serviços na *Web*. Em Junho de 2001 foi descrita uma especificação sobre a arquitectura que os Serviços *Web* deveriam seguir, de forma a permitir uma maior flexibilidade (UDDI v2.0). Em 2002 saiu a especificação UDDI v2.0 através da OASIS. Em Julho de 2003, o UDDI v3.0 foi considerado o suporte de interacção de implementação privada e pública. Em 2004 saiu como especificação UDDI v3.0 através da OASIS. [UDDI 2006]

O UDDI baseia-se nos padrões definidos pelo W3C e IETF (Internet Engineering Task Force), tais como o XML, o HTTP e o DNS. Adicionalmente, as características de programação das plataformas são suportadas, adoptando-se versões do SOAP conhecidas como as especificações de mensagens XML. [UDDI 2004]

2.1.3.3.2 Estrutura do UDDI

O servidor UDDI disponibiliza uma Base de Dados para armazenar informações sobre Serviços *Web*. Estas informações encontram-se replicadas em várias bases de dados, visto existirem várias empresas (MICROSOFT, IBM, SAP, ORACLE, entre outras) a suportarem este tipo de serviço. Desta forma, uma aplicação que pretenda registar um *Web Service* apenas tem de se registar num único servidor, não havendo necessidade de o fazer em cada um dos vários servidores UDDI disponíveis. Mas o servidor UDDI não é apenas uma Base de Dados onde constam as ditas informações, o UDDI disponibiliza ainda três serviços básicos:

- **Registo –** este serviço descreve como a entidade **Fornecedora** de serviços se regista e regista os seus *Web Services*;
- **Pesquisa –** este serviço descreve como uma aplicação **Cliente** encontra a especificação do *Web Service* e/ou do seu **Fornecedor**;
- **Mapeamento –** este serviço descreve a forma como uma aplicação **Cliente** invoca e interage com um *Web* Service após a sua localização.

O UDDI utiliza o *"*XML *Schema"* [XML Schema 2004a, XML Schema 2004b, XML Schema 2004c] para descrever formalmente as suas estruturas de dados. Estas estruturas, resultantes do processo de registo, são bem definidas e permitem realizar uma posterior pesquisa de informações específicas. A pesquisa é feita pelo tipo de serviços, funcionando um pouco como as páginas amarelas dos Serviços *Web*. Assim, a informação guardada no directório do UDDI pode ser apresentada em três tipos de categorias:

- Páginas Brancas (*White Pages*);
- Páginas Amarelas (*Yellow Pages)*;
- Páginas Verdes (*Green Pages*).

Estas informações encontram-se no directório do UDDI organizadas como se mostra na Figura 15.

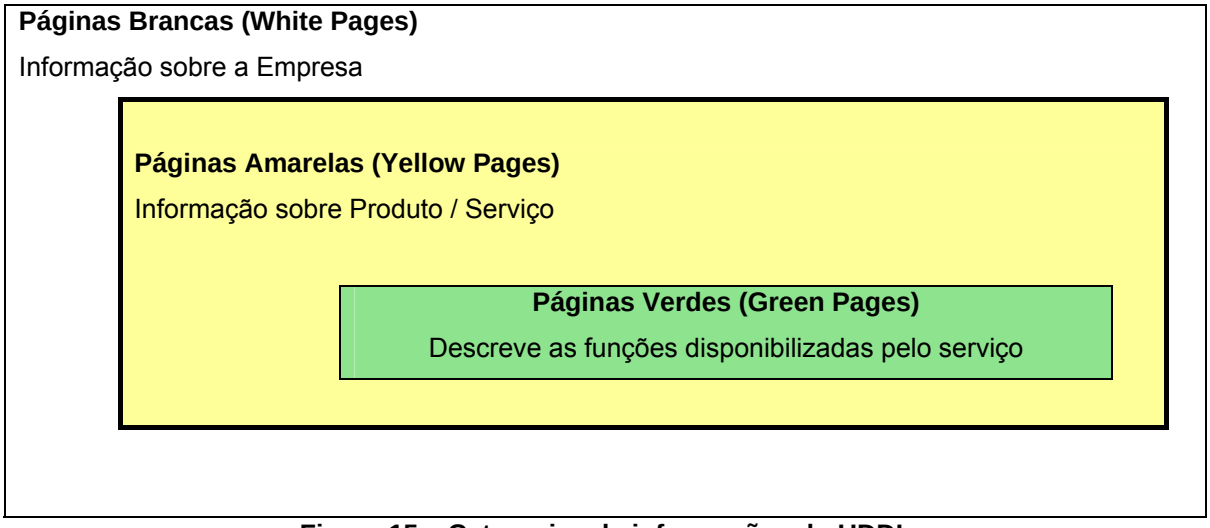

Figura 15 – Categorias de informações do UDDI.

Páginas Brancas *(White Pages)* **–** estrutura do directório UDDI contêm a identificação básica da empresa que fornece o *Web Service:*

- Nome;
- Morada;
- Telefone;
- Descrição da empresa
- etc...

É também possível encontrar lista de identificadores alternativos para identificar a empresa.

- ***Páginas Amarelas*** *(Yellow Pages)* **–** estrutura do directório UDDI que contém a informação quer do *Web Service* quer do seu **Fornecedor**, registados por categorias em que se inserem:
    - Tipo de serviço;
    - Localização geográfica.
- ***Páginas Verdes*** *(Green Pages)* **–** estrutura do directório UDDI que contêm informação técnica que descreve as funções disponibilizadas pelo serviço, encontram-se também informações sobre a localização do *Web Service*(localização do documento WSDL ou endereço URL).

O registo completo de um *Web Service* no UDDI consiste no preenchimento das informações necessárias a estas três categorias (*White Pages, Yellow Pages e Green Pages*).

A especificação UDDI inclui um XML *Schema* que define quatro tipos de informação principais, estando eles relacionados.

Na entidade de negócios *businessEntity* é possível descrever a informação relativa ao **Fornecedor** do serviço, correspondendo à informação contida nas Páginas Brancas (*White Pages)* anteriormente referidas. A estrutura do *businessEntity*, segundo a UDDI *Data Structure Reference V1.0*, encontra-se exemplificada na Figura 16.

```
<element name = "businessEntity">
    <type content = "elementOnly">
        <group order = "seq">
            <element ref = "discoveryURLs" minOccurs = "0" maxOccurs = "1" />
            <element ref = "name" />
            <element ref = "description" minOccurs = "0" maxOccurs = "*" />
            <element ref = "contacts" minOccurs = "0" maxOccurs = "1" />
            <element ref = "businessServices" minOccurs = "0" maxOccurs = "1" />
            <element ref = "identifiedBag" minOccurs = "0" maxOccurs = "1" />
            <element ref = "categoryBag" minOccurs = "0" maxOccurs = "1" />
        </group>
          <attribute name = "businessKey" minOccurs = "1" type = "string" />
          <attribute name = "operator" type = "string" />
          <attribute name = "authorizedName" type = "string" />
      </type>
  </element>
```

Figura 16 – Estrutura 'businessEntity' do UDDI.

O campo *name* fornece o nome da entidade **Fornecedora** do *Web Service*, ou seja, a empresa que está a realizar o registo.

No campo *description* pode ser armazenada uma breve descrição do **Fornecedor** *Web Service*.

No campo *contacts* pode ser guardada a informação relativa aos contactos da entidade que disponibiliza o serviço.

No campo *businessServices* pode ser guardada uma lista de todos os businessServices disponibilizados pela entidade **Fornecedora** do serviço.

No campo *identifiedBag* encontra-se uma lista de identificadores alternativos para o **Fornecedor** do *Web Service*.
No campo *categoryBag* encontra-se uma lista de pares nome/valor, que identificam o **Fornecedor** do serviço de acordo com a categoria pela qual o *Web Service* pode ser procurado.
Da estrutura acima descrita, destaca-se o *businessKey* como sendo a chave do registo UDDI. Esta chave, em formato hexadecimal, permite definir univocamente o **Fornecedor** do serviço e é-lhe atribuída pelo UDDI recorrendo a um algoritmo denominado UUID *(Universal Unique Identifier*).
O campo *operator* identifica o nome do UDDI onde o registo foi efectuado. Se o registo tiver sido efectuado na MICROSOFT, este campo terá a informação HTTP://UDDI.microsoft.com.
No campo *authorizedName* encontra-se o nome de quem regista as informações relativas à entidade **Fornecedora** do serviço.
Na estrutura *businessService*, encontram-se as informações sobre cada um dos serviços oferecidos pela empresa, correspondendo à informação encontrada nas Páginas Amarelas (*Yellow Pages)*. Para cada *businessEntity* podem existir vários *businessService*, sendo este relacionamento feito através do campo *businessKey*. A estrutura do *businessService*, segundo a UDDI *Data Structure Reference V1.0*, é a que se apresenta como exemplo na Figura 17.

```
<element name = "businessService">
    <type content = "elementOnly">
        <group order = "seq">
            <element ref = "name" />
            <element ref = "description" minOccurs = "0" maxOccurs = "*" />
            <element ref = "contacts" minOccurs = "0" maxOccurs = "1" />
            <element ref = "bindingTemplates" />
            <element ref = "categoryBag" minOccurs = "0" maxOccurs = "1" />
        </group>
        <attribute name = "serviceKey" minOccurs = "1" type = "string" />
        <attribute name = "businessKey" type = "string" />
    </type>
</element>
```

Figura 17 – Estrutura *businessService* do UDDI.

O campo *name* contém o nome do serviço em causa.
O campo *description* contém uma descrição das funcionalidades do serviço.
O campo *bindingTemplates* contém uma lista dos diversos *bindingTemplates* disponíveis no serviço.
No campo *categoryBag* encontra-se uma lista de pares nome/valor, que identificam o serviço de acordo com a categoria pela qual pode ser procurado.
No campo *serviceKey* encontra-se um atributo que permite definir univocamente o *Web Service*.
No campo *businessKey* consta a chave da *businessEntity* que contém o serviço.
Na estrutura *bindingTemplate* (encontram-se descritas as informações técnicas sobre o *Web Service*, correspondendo à informação encontrada nas Páginas Verdes (*Green Pages*). A estrutura do *bindingTemplate*, segundo a UDDI *Data Structure Reference V1.0*, é a que se apresenta como exemplo na Figura 18.

```
<element name = "bindingTemplate">
    <type content = "elementOnly">
        <group order = "seq">
            <element ref = "description" minOccurs = "0" maxOccurs = "*" />
                <group order = "choice">
                    <element ref = "accessPoint" minOccurs = "0" maxOccurs = "1" />
                    <element ref = "hostingRedirector" minOccurs = "0" maxOccurs = "1" />
                </group>
                <element ref = "tModelInstanceDetails" />
        </group>
        <attribute name = "bindingKey" minOccurs = "1" type = "string" />
        <attribute name = "serviceKey" type = "string" />
</type>
</element>
```

Figura 18 – Estrutura do *bindingTemplate* do UDDI.

No campo *description* encontra-se uma descrição do *bindingTemplate.*

No campo *accessPoint* encontra-se a localização do *Web Service*. Normalmente é um endereço URL, mas também pode ser um endereço e-mail ou número de telefone. O *accessPoint* possui um atributo *urlType* que indica o tipo de informação a que se refere (mailto, HTTP, HTTPs, ftp, fax, phone, etc.).

No campo *hostingRedirector* só existe informação quando não é definido nenhum *accessPoint*, passando assim a referenciar outro bindingTemplate**.**

No campo *tModelInstanceDetails* pode ser encontrada uma lista de elementos *tModelInstanceInfo*, os quais contêm referências para elementos *tModel*. Essas referências são feitas através do atributo obrigatório *tModelKey*, o qual identifica univocamente o *tModel* ao qual se pretende referir.

Finalmente, na estrutura *tModel* pode ser encontrada a localização da informação técnica do *Web Service* em causa. Na realidade, esta é utilizada sempre que é necessário indicar uma descrição que se encontra num ficheiro externo. A estrutura do *tModel*, segundo a UDDI *Data Structure Reference V1.0*, é a que se apresenta como exemplo na Figura 19.

```
<element name = "tModel">
    <type content = "elementOnly">
        <group order = "seq">
            <element ref = "name" />
            <element ref = "description" minOccurs = "0" maxOccurs = "*" />
            <element ref = "overviewDoc" minOccurs = "0" maxOccurs = "1" />
            <element ref = "identifierBag" minOccurs = "0" maxOccurs = "1" />
            <element ref = "categoryBag" minOccurs = "0" maxOccurs = "1" />
        </group>
        <attribute name = "tModelKey" minOccurs = "1" type = "string" />
        <attribute name = "operator" type = "string" />
        <attribute name = "autorizedName" type = "string" />
    </type>
</element>
```

Figura 19 – Estrutura do *'tModel'* do UDDI.

O campo *overviewDoc* contém a referência para a localização do ficheiro externo onde se encontra a especificação do *Web Service*. Para tal conta-se com o sub-elemento *overviewURL* que indica a localização do ficheiro propriamente dito (o URL ou o WSDL).

Assim, tendo em conta as ligações entres as várias estruturas de dados do UDDI, pode-se construir o seguinte esquema de relacionamentos.

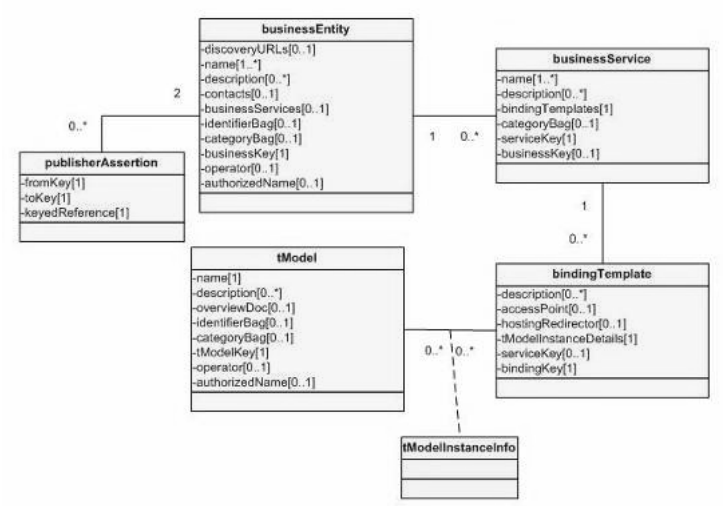

Figura 20 – Esquema de relacionamentos entre as estruturas do UDDI.

2.1.3.3.3 Métodos do UDDI

Independentemente das plataformas ou linguagens de programação usadas, a fase de Publicação ou Divulgação consiste em criar, ler, actualizar ou apagar as informações relativas aos **Fornecedores** e aos Serviços *Web* existentes no directório UDDI. Assim podem ser chamadas as seguintes funções durante o processo de registo ou publicação (Tabela 1 **– Funções do UDDI na fase de Publicação**). [DEV 2005]

Tabela 1 – Funções do UDDI na fase de Publicação

Método	**Descrição**
Delete_binding	Remove um ou mais registos *bindingTemplate* do directório UDDI
Delete_business	Remove um ou mais registos *business* do directório UDDI
Delete_service	Remove um ou mais registos *businesService* do directório UDDI
Delete_tModel	Remove um ou mais registos *tModel* do directório UDDI
save_binding	Grava ou actualiza o elemento *bindingTemplate*
save_business	Grava ou actualiza informações respeitantes à estrutura *businessEntity*
save_service	Grava ou actualiza o elemento *businessService*
save_tModel	Grava ou actualiza o elemento *tModel*

Durante a fase de pesquisa ou invocação são realizadas as operações de pesquisa de **Fornecedores** ou dos serviços específicos. Na Tabela 2 podem ser observados alguns dos comandos utilizados para realizar estas operações.

Tabela 2 – Funções do UDDI na fase de Invocação

Método	**Descrição**
find_binding	Descobre 'bindings' existentes nos 'businessServices'
find_business	Localiza informações relativas aos *businesses*
find_relatedBusinesses	Localiza informações relativas aos registos *businessEntity***,** registos estes que se encontram relacionados com um *business entity* específico cuja *key* foi obtida numa consulta
find_service	Descobre *services* específicos registados no *business entities*
find_tModel	Descobre as informações respeitantes aos *tModel* existentes na estrutura
get_bindingDetail	Recolhe informações do *bindingTemplate*
get_businessDetail	Recolhe informações *businessEntity*
get_businessDetailExt	Recolhe informações adicionais do *businessEntity*
get_serviceDetail	Recolhe informações do *businessService*
get_tModelDetail	Recolhe informações do *tModel*

2.1.4 Sistemas flexíveis, virtuais e dinâmicos.

Nesta secção ira ser abordado o tema dos sistemas flexíveis, virtuais e dinâmicos, o *Sistema Ubíquo de Computação*, Empresas e Sistemas flexíveis, virtuais e dinâmicos de Produção.

2.1.4.1 Sistema Ubíquo de Computação

Com o desenvolvimento de tecnologias cada vez mais transparentes, têm aparecido novos conceitos para definir nichos de mercado e se estabelecerem como as bases de pesquisas que estão em desenvolvimento, para serem naturalmente parte do nosso dia-a-dia no futuro.

Exemplos mais marcantes disso são os dispositivos portáteis multifuncionais – que vêm substituir as agendas, telefones ou mesmo o computador pessoal – até as ainda distantes redes *ad hoc* de micro sensores com capacidade de processamento e comunicação e propiciarem a criação de estruturas de comunicação em ambientes até então impraticáveis, a *smart dust,* alvo de pesquisas em universidades norte-americanas e com financiamento da agência DARPA (*Defense Advanced Research Projects Agency*), do governo dos EUA. (Stajano, 2001).

A área de pesquisa que sugere novas formas de interacção entre utilizador e máquina é a *Computação Ubíqua*. A *Computação Ubíqua* também designada por "*pervasive*

Computing"(computação patente) ou "*calm technology*" (tecnologia da calma), tem suas origens no trabalho seminal de Mark Weiser em Xerox Palo Alto no ano de 1988.

Segundo o mesmo Mark Weiser, a primeira era da computação foi relacionada aos *mainframes* onde vários usuários estavam conectados a uma única estação de trabalho. A segunda era da computação iniciou com a popularização dos computadores e o surgimento dos computadores pessoais (PC – *Personal Computers*). Hoje estamos em uma fase de transição entre a segunda era da computação e a Computação Ubíqua, tida como terceira grande era da computação

Existem várias definições para C*omputação Ubíqua*, passo a indicar algumas:

A *Computação Ubíqua* consiste em desenvolver ambientes onde a introdução de tecnologia no quotidiano ocorra de modo natural e transparente. Nesses ambientes, ditos ubíquos, existe uma enorme presença de mobilidade e informações como identificação, actividade, preferências e histórico de utilizadores são monitoradas, processadas e armazenadas por diferentes dispositivos e aplicações.

Computação Ubíqua é a capacidade de estar conectado à rede e fazer uso da conexão a todo o momento.

O conceito básico da *Computação Ubíqua* é de que vários dispositivos, comunicantes entre si, estejam conectados a um único usuário fornecendo diversos serviços computacionais. O tamanho dos dispositivos computacionais cada vez mais reduzido e suas capacidades de processamento e armazenamento cada vez maiores são evoluções atuais que possibilitarão a criação de ambientes ubíquos. Outro factor que contribui muito com a concretização da Computação Ubíqua é a evolução das tecnologias de comunicação das redes de computadores.

Com o aparecimento do computador ENIAC[2] em 1943, que pesava cerca 30 toneladas, consumia 200 000 watts de potência e ocupava várias salas, Iniciou-se a era dos computadores. Nessa época havia um grande computador para servir várias pessoas, como era o caso de ENIAC que era usado pelo exército norte-americano, em plena II Guerra Mundial, tendo como objectivo o auxilio nos cálculos de precisão necessários para a balística.

[2] *Electronic Numerical Integrator and Computer* - O primeiro computador do mundo em 1943, realizado pelos Professores John Mauchly e J. Presper Eckert.

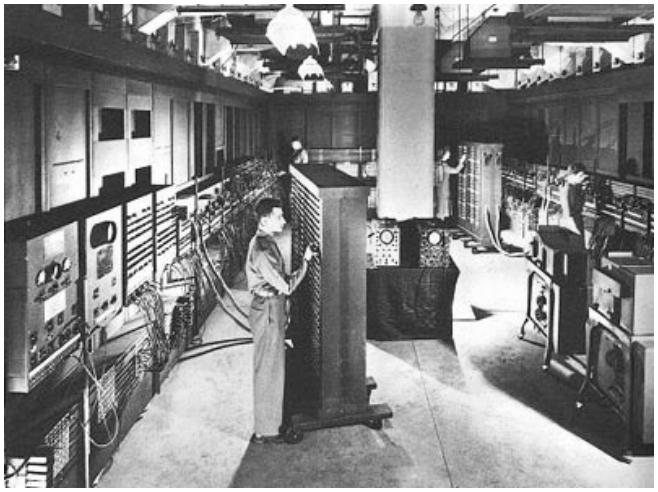

Figura 21 - ENIAC (*Electronic Numerical Integrator and Computer*).

Com a evolução constante a que se assistiu no final do século passado, fez com que em meados da década de 70 os computadores começassem a ter preços cada vez mais acessíveis e em 1981 a IBM lançou no mercado o PC. Nessa altura começou-se a alterar a tendência do número de computadores por pessoa, a partir desse altura começou a haver cada vez mais computadores e hoje em dia há uma grande proliferação dos dispositivos inteligentes que nos cercam, hoje vivemos numa era em que cada pessoa tem vários computadores.

Como hoje em dia cada pessoa tem vários computadores e outros dispositivos de comunicação, existe a necessidade de se poder partilhar a informação pelos vários dispositivos de computação, assim pode-se afirmar que a computação se está tornada rápida e ubíqua, visto podemos aceder ao computadores e a informação em qualquer parte e a qualquer hora.

Hoje em dia existem muitos especialistas que estão a desenvolver trabalhos na área da *Computação Ubíqua* com o objectivo de criar uma rede ubíqua sem fio, tirando para isso o melhor partido das tecnologias de sistemas microeletromecânicos (MEMS) e dos sistemas de RF (radio-frequência).

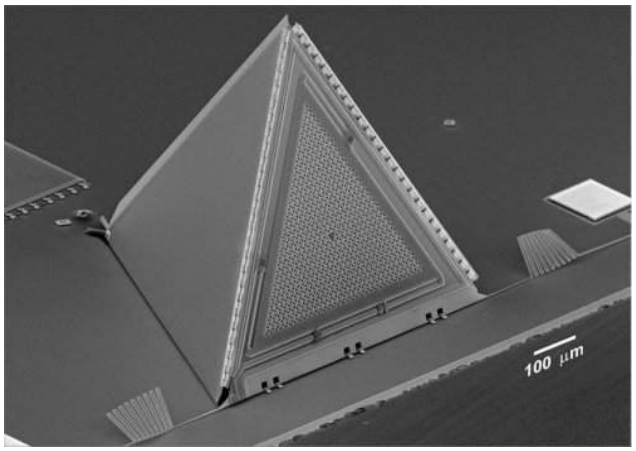

Figura 22 – Exemplo de um sistema microeletromecânico.

O aparecimento das redes s*elf-organizing* de sensores e dos actuadores autónomos, faz com que haja um aumentando da flexibilidade e da reconfiguração dos sistemas de produção e fabrico, conseguindo-se assim sistemas ubíquos totais.

Em Arquitecturas Orientadas de Serviços, os dispositivos da rede devem ser capazes de reconhecer um novo dispositivo, atribuir-lhe um endereço, descobrir os serviços que este pode fornecer aos outros dispositivos, obter uma descrição das suas funcionalidades, enviar-lhe alguns pedidos de acordo com a descrição do mesmo e ser capaz de subscrever eventos desse dispositivo. O dispositivo deverá ter um ficheiro .XML que permita a um controlador obter informações sobre a sua descrição e a forma como controlador deve usar serviços disponibilizados por esse dispositivo. O ponto mais relevante é o facto destes novos dispositivos poderem ser integrados na rede, e os seus serviços poderem ser interligados com outros serviços já existentes, para assim seriar serviços de mais alto nível (*holonic system*[3]).

Como todos os dispositivos da rede necessitam trocar informações para trabalhar de forma autónoma, a conectividade é um factor crucial na *Computação Ubíqua*. A conectividade é a garantia do bom funcionamento do ambiente ubíquo desde a troca de informações entre os dispositivos. Um alto grau de pervasividade[4] é necessário em ambientes ubíquos. Isso é devido ao facto de um ambiente ubíquo dever ser pensado como uma rede de comunicação com diversos dispositivos Wireless e, em muitas casos, móveis.

Um exemplo de um sistema interligado para criar serviços de mais alto nível, é o caso dos sensores de nível, pois são os dispositivos que nos dão informação sobre o nível de líquido contido num reservatório, essa informação ao ser enviada para um dispositivo de mais alto nível, como o pode ser o caso de um tapete de um transportador, que por sua vez, pode ser uma peça de um linha de enchimento. Essa linha interligada com outros dispositivos da linha de produção, podem então dar forma a um sistema flexível de produção (FMS – *Flexible Manufacturing System*).

3 A *holonic system* é composto por entidades autónomas (designados por *holons*), que podem deliberadamente reduzir a sua autonomia, quando necessário, para atingir um objectivo colectivo.

4 Computação pervasiva: responsável pela adaptação dinâmica de um dispositivo a um determinado ambiente.

2.1.4.2 Empresas e Sistemas flexíveis, virtuais e dinâmicos de Produção

Ao termos um conjunto de dispositivos inteligentes e flexíveis como os acima referidos, podem-se ter sistemas mais dinâmicos e reconfiguráveis automaticamente para execução de novas tarefas, dizendo-se então que estamos perante um *Sistema de Produção flexível, virtual e dinâmico*.
Uma empresa ao empregar este tipo de sistema de produção irá verificar que este sistema lhe permitirá adaptar e melhorar o serviço de acordo com as suas necessidades, visto este sistema ter capacidade de decisão própria e rapidez para adaptar-se a novas situações, podendo ser estas devido a falhas de matéria prima, avarias de máquinas ou imprevistos de outro tipo.
De acordo com o Mark Weiser a *Computação Ubíqua* prevê que a longo prazo o computador pessoal e a estação de trabalho poderão perder importância devido ao facto de o acesso à computação estará em toda os locais, logo poderemos ter acesso à informação em todos os lugares.
A Tecnologia de Computação evoluiu até ao ponto em que o Sistema de *Computação Ubíqua* se tornou possível usando os dispositivos de rede actuais bem como protocolos e aplicações.
Modelos e sistemas estão ser desenvolvidos de maneira a conseguir a aplicação deste conceito de *Sistema de Produção flexível, virtual e dinâmico*. Este conceito está relacionado com a disponibilidade de controlo das operações de produção estarem em toda a parte, usando para isso o controlo directo, computadores pessoais, computadores portáteis e PDA's.
Algumas das actividades que devem ser executadas são:

1. o projecto dos modelos para representar a informação de **produto**, a informação da gestão de **produtos**, os processos e os recursos;
2. a integração entre as funções do projecto (CAD/CAPP) e a fabricação do computador controle e operação (CAM) com as funções do planeamento e do controle da produção(PPC).

De acordo com *"BM_Virtual Enterprise Architecture Reference Model*", que define um sistema de produção ou empresa virtual como um sistema de controlo multi-nível sobre a rede da empresa, assegurando a integração, a distribuição, a agilidade e a virtualidade do sistema. As exigências mencionadas são as mesmas para um sistema de produção flexível, virtual e dinâmico.
Distribuição é relacionado com a distribuição geográfica dos elementos do sistema, de que deve ser interligada, assegurando a operação e o controle a partir de algum ponto do mundo (ou seja "fabricação global "), promovendo a internacionalização do negócio.
A agilidade significa que o sistema pode ser reconfigurado rapidamente e este será um factor importante para a concorrência das empresas, assim as empresas não vão ficar dependentes das encomendas de um grupo reduzido de **Fornecedores**. Este é o caso de grande parte das empresas metalomecânicas que fornecem componentes metálicos para a industria automóvel. Este tipo de industria são um tipo de industria tipicamente de produção em série, só que com a

evolução que tem havido na industria automóvel, que com o constante lançamento de novos modelos, que leva ao desenvolvimento e criação de novas peças, faz com que as essas empresas metalomecânicas que dantes faziam grandes séries de peças, sendo em certos casos a mesma peça durante anos, tenham de se adaptar e alterar os seus modos de produção para assim se poderem reconfigurar rapidamente às novas peças. Esta alteração passa pelo uso da subcontratação, uso das novas tecnologias de modo a que se tenha um sistema de produção cada vez mais flexível e dinâmico.

Com o desenvolvimento das comunicações computadorizadas em rede, tornou-se corrente os termos "virtual" e "virtualidade". Geralmente, chama-se "virtual" a tudo aquilo que diz respeito às comunicações via Internet, embora essa conotação da palavra não seja a mais correcta, porque "virtual" implica o conceito de uma simulação, o que nem sempre é verdade.

O Virtualidade é também relacionada com ambientes virtuais como “a presença virtual”, a realidade virtual e a realidade aumentada. Desta maneira o *Sistemas de Produção e Empresas flexíveis, virtuais e dinâmicas* fornecem a produção e os serviços a empresas a qualquer momento e em qualquer lugar.

Segundo os autores o artigo “*Towards Ubiquitous Production Systems and Enterprises*” os elementos atrás apresentados são a base para a construção de um *Sistema de Produção Ubíquo.*

O conceito de *Sistemas de Produção e Empresas Ubíquas* (*Ubiquitous Poroduction System and Enterprises* - UPSE) está baseado em Sistemas e Empresas de Produção Distribuídas e Virtuais e o conceito de Sistemas de Produção e Empresas Ubíquas tem uma grande envolvência da *Computação Ubíqua*.

2.1.5 Considerações Sobre as Tecnologias de Suporte

Nesta secção vão ser feitas algumas considerações sobre as tecnologias de suporte atrás descritas, nomeadamente o DCE, CORBA e *Web Services* (WSDL, UDDI e SOAP), como ferramentas/utilitários para dar respostas aos problemas inicialmente referidos no início nesta dissertação. Assim, pretende-se avaliar como estas tecnologias podem suportar a solução proposta no capítulo 1, durante a localização dos Servidores *Web* (**Fornecedores**) e durante a identificação e a invocação dos seus serviços (**Produtos**).

O CORBA e o DCE foram concebidos para permitir que as aplicações distribuídas sejam independentes do hardware, dos sistemas operativos e das linguagens de programação.

As aplicações distribuídas que utilizam os *Web Services* são também independentes do hardware e dos sistemas operativos. As aplicações distribuídas que utilizem os *Web Services* apenas necessitam de enviar e receber mensagens do tipo SOAP, utilizando para o efeito diversos protocolos de comunicação, nomeadamente os protocolos HTTP/TCP/IP.

Em SOAP/XML podem ser propostas “etiquetas” para dar resposta a novas necessidades, originadas por novas situações, ao passo que as plataformas distribuídas DCE e CORBA não são

tão acessíveis. Um novo método ou função implica um registo deste no Directório de Serviços DCE ou IR (CORBA).
O CORBA é um componente orientado para objectos enquanto que os Serviços *Web* e o DCE se preocupam mais com a mensagem em si, abstraindo-se da noção de objectos.
O DCE suporta *contexts* que são mecanismos que mantêm o estado do servidor durante o pedido de um **Cliente**, isto é, o **Cliente** pode estar a transmitir uma série de RPC para realizar uma operação ou acção individualmente. Estes *contexts* são geridos directamente e automaticamente pelo *Stub's* do **Cliente** e do servidor. O CORBA não possui este mecanismo, sendo o programador responsável pela gestão da transmissão e recepção dos RPCs.

2.1.5.1 Localizar Servidores

Nesta fase pretende-se analisar o comportamento das tecnologias de suporte quando é necessário localizar um computador, existente numa rede distribuída, onde reside uma aplicação solicitada por um **Cliente**. As aplicações **Cliente** devem ser capazes de interagir com outras aplicações existentes numa rede distribuída de forma a executarem os serviços pretendidos, independentemente de conhecerem ou não as suas localizações. Devem também possuir mecanismos capazes de identificar e localizar esses serviços ou objectos.
Existe um componente no CORBA, chamado Directório de Implementação (*Interface Repository*), que contém as informações necessárias para permitir ao ORB localizar e processar um pedido de um determinado objecto a um servidor, mas apenas pelo nome que foi inicialmente identificado na rede. Nos Serviços *Web*, o UDDI disponibiliza uma Base de Dados na *Web* onde os serviços podem ser localizados pelo nome da entidade, pelo tipo de actividade ou pelo seu URL. Mostra-se assim uma forma mais completa de descobrir os serviços existentes e que podem ser invocados.

2.1.5.2 Identificar Serviços Fornecidos por Servidores

O CORBA possui um Directório de Implementação onde constam as informações respeitantes às operações que podem ser executadas num determinado objecto. A invocação desse objecto é feita usando o DII (*Dynamic Invocation Interface*).
No DCE é possível identificar os serviços no chamado directório de serviços DCE. A capacidade de localizar o serviço através deste directório é da responsabilidade do RPC DCE. Um **Cliente** pode procurar um serviço pela sua proximidade, pelo seu UUID (*Universally Unique Identifier*), pelo nome ou até pelo seu tipo. No entanto, este repositório de informação pode não contemplar todos os serviços existentes na rede. Nos Serviços *Web* é utilizada a especificação WSDL para definir na forma de um documento os serviços disponíveis, as variáveis utilizadas, as formas de interacção com o servidor que possui esse serviço, o protocolo de comunicação, etc. Este serviço

mostra-se mais flexível no que diz respeito à introdução de novas funções ou tipos de dados a utilizar.

2.1.5.3 Invocar Serviços num Servidor

As aplicações que usam o RPC em sistemas computacionais distribuídos anteriores ao aparecimento da Internet, casos do DCE e do CORBA, podem encontrar problemas de compatibilidade e segurança durante a transmissão de dados quando se tentam adaptar aos novos meios de transmissão. Foi referido anteriormente que estas especificações possuem componentes que formatam automaticamente a mensagem a transmitir para a rede (*Stub's* no **Cliente** e no servidor). Estes problemas centram-se essencialmente no bloqueio da transmissão de dados por parte de *firewalls* e *proxys* existentes nos servidores. O uso da Internet, mais propriamente através do protocolo HTTP, é um excelente e o mais utilizado meio de transmissão de mensagens entre aplicações nos dias de hoje, visto o HTTP ser suportado pela maior parte dos Browser *Web* e servidores.

Os *Web Services* já utilizam a Internet (HTTP) para solicitarem serviços aos servidores e receberem respostas destes, através de mensagens SOAP. Estas mensagens podem ser à primeira vista trabalhosas e por vezes complexas a criar, mas no final conseguem mostrar de uma forma clara e simples os parâmetros que se querem passar para a rede. Além disso, podem ser facilmente apresentadas nos Bro*Web Service*ers *Web* tendo que sofrer para isso uma simples transformação para HTML (*HyperText Markup Language*). Além do HTTP, as mensagens também podem ser transmitidas para a rede utilizando protocolos de comunicação como SMTP, e-mail, entre outros.

Tabela 3 – Comparação entre as tecnologias de suporte

	CORBA	**DCE**	***Web* Services**
Sistema	IDL (estático + verificação em tempo real)	IDL (dinâmico + verificação em tempo real)	XML Schemas (verificação em tempo real)
Sintaxe Transferência	CDR (formato binário)	CDR (formato binário)	XML (UTF-Unicode)
Estado	Statefull	Stateless	Stateless
Registo	Interface Repository Implementation Repository	Directório de serviços DCE	UDDI/WSDL
Serviço de localização	CORBA naming/trading service	Através UUID's no directório de serviços DCE	UDDI
Segurança	Plataforma CORBA	Plataforma DCE (Kerberos)	Assinaturas HTTP's, XML
Firewall	Não utilizado em HTTP	Não utilizado em HTTP	Através HTTP
Criação Mensagens	CORBA Stub's	RPCs Stub's	SOAP
Gestão erros	Excepções IDL	ACF (Attribute Configuration File) CATCHANY,CATCH ALL	Mensagens Erro SOAP
Modelo Data	Objectos	Procedimentos	Mensagens SOAP
Ligação *Cliente*/Servidor	Directa	Directa	Indirecta
Parâmetros passados	Por referência e por valor	Pelo UUID	Por valor (não é tido em conta o objecto)
Meio Transmissão	GIOP/IIOP/TCP/IP	GIOP/IIOP/TCP/IP	HTTP/SMTP/TCP/IP
Linguagem Programação	Todas as que usem IDL	Todas as que usem IDL	Todas as que interpretem SOAP ou XML

2.2 Propostas de arquitecturas e serviços que impliquem os serviços de *Broker*

Vários trabalhos nesta área e em áreas afins têm sido desenvolvidos por diversos autores. Nesta secção são apresentados vários trabalhos e dissertações sobre esta temática, tal como os seus objectivos e as conclusões obtidas, de uma forma resumida. No final falar-se-á do caso de sucesso da Amazon *E-Commerce*.

Uma dissertação de mestrado intitulada "*Web* Service Interface and Architecture for Accessing FieldBus System" propõe um conjunto de serviços *Web* para que seja possível aceder a uma rede de campo a partir da *Web*, mais exactamente, ler e escrever valores em diversos dispositivos electrónicos ligados a essas redes de campo. O autor recorre também às normas WSDL para definir os serviços *Web* que propõe e à norma SOAP para definir as operações e o formato das mensagens trocadas. Como conclusão, o autor refere que a solução por si proposta permite um elevado nível de abstracção dos dispositivos da rede de campo, da estrutura de dados e do seu formato. Refere ainda que uma arquitectura/interface *Web* baseada em HTTP, SOAP pode ser facilmente acessível por outros sistemas e o código desenvolvido pode ser reutilizado com facilidade em diferentes plataformas computacionais. [Merino 2004]

Um trabalho de fim de curso intitulado "Disponibilização e tratamento de informação na Internet orientados à interoperabilidade utilizando XML", apresenta a *Extensible Markup Language* - XML e as tecnologias relacionadas a esta linguagem como uma alternativa a *Hypertext Markup Language* - HTML como uma forma de publicar informação na Internet. Demonstra dois casos práticos onde esta ideia pode ser utilizada, um breve estudo sobre as tecnologias XML e sobre *Web Services* é feito, fornecendo a base necessária para entender o conceito aqui apresentado. [Gomes 2005]

Outra dissertação de mestrado intitulada "Disponibilização de serviços baseados em localização via *Web Services*". Esta tese propõe a adopção da tecnologia *Web Services* e a utilização de padrões abertos na construção de soluções LBS(*Location-Based Services)*. A proposta de disponibilização de aplicações LBS via *Web Services* foi validada através do desenvolvimento de um protótipo envolvendo dois serviços que integram o projecto SAM: Serviço de Apresentação de Mapas e Serviço de Localização. Este protótipo permite a visualização da localização de um determinado dispositivo móvel. No caso de uso em questão, a aplicação **Cliente** solicita ao Serviço de Localização a posição geográfica (X,Y) de um determinado dispositivo móvel. Com base na coordenada obtida, o **Cliente** solicita ao Serviço de Apresentação a geração de um mapa com a localização do dispositivo. [Silva, Grace ; Pereira, Patrícia e Magalhães Geovane, 2004]

Outra dissertação de mestrado intitulada "Um estudo sobre o desenvolvimento orientado a serviços", esta dissertação discute o que realmente o desenvolvimento orientado a serviços apresenta como novidade tecnológica, através da discussão de todas suas características consideradas relevantes para construção de uma aplicação baseada no conceito de serviços.

Características como re-uso, caixa-preta, distribuição e suporte a heterogeneidade ambiental, entre outras, são discutidas no intuito de identificar suas vantagens e funcionalidades, e principalmente, sua importância para o desenvolvimento orientado a serviços. É discutido, também, o conceito representado pelo termo "serviço", que possui interpretações distintas na indústria e em publicações académicas. São apresentados também alguns frameworks que dão suporte ao desenvolvimento orientado a serviços, como Vinci, Jini e os XML *Web Services*. Finalmente, é apresentada uma proposta para um novo framework que oferece suporte ao desenvolvimento orientado a serviços, e que tem como diferencial a opção de se basear na extensão da infra-estrutura de um servidor de aplicações J2EE.[Machado, João 2004]

Outra dissertação de mestrado intitulada "*Web*Graf – Aplicação *Web* para Execução de Grafcets e redes de Petri em Controladores Lógicos Programáveis" propõe uma ferramenta de tradução automática de GRAFCET ou redes de Petri para Lista de Instruções. Esta ferramenta está disponível na *Web* e pode ser acedida a partir de um vulgar Browser *Web*. Os programas escritos em GRAFCET ou Petri são previamente convertidos para um documento XML, usando um conjunto de "etiquetas XML", por ele propostas. Esse documento é então enviado para o servidor *Web* onde reside a ferramenta de tradução proposta. A partir deste documento a ferramenta de tradução gera um programa autómato em lista de instruções. O algoritmo de conversão foi escrito em JSP (Java Server Pages). [Gomes 2003]

Outra dissertação de mestrado intitulada "Identificação e Controlo de Processos via Internet" propõe uma arquitectura que permite "a utilização remota de laboratórios pelos alunos, mesmo a partir de casa, sem preocupações de horário e sem preocupações com o eventual excesso de alunos que possam estar presentes no laboratório". Remotamente os alunos podem testar diversos algoritmos de controlo e observar o comportamento físico de vários processos no laboratório. Para permitir que um determinado sistema físico possa ser monitorizado e controlado remotamente via Internet, a arquitectura proposta utiliza um servidor *Web*, um servidor de correio electrónico SMTP e um servidor de FTP, além de tecnologias COM, DDE, DAO, ADO para aceder aos recursos laboratoriais e às bases de dados para registar os resultados das experiências. Foi possível ao autor concluir que esta solução permite que o controlo de sistemas pode ser ensinado recorrendo à utilização remota de laboratórios. [Silva 2003]

Outra dissertação de mestrado intitulada "Estudo e construção de um ambiente de grade computacional peer-to-peer com ênfase no balanceamento de carga", tem como objectivo processar aplicações que executam grandes quantidades de cálculos, exigindo recursos com grandes capacidades de processamento, são os responsáveis por estudos voltados à criação de sistemas computacionais de alto desempenho, várias soluções são utilizadas, as quais dividem-se basicamente em sistemas compostos por máquinas multiprocessadoras paralelas e sistemas de computação paralela e distribuída.

Esta tese de mestrado propõe-se a definir uma arquitectura de *software* composta por um conjunto de mecanismos de gerência de recursos distribuídos, capaz de distribuir tarefas entre eles segundo um critério de balanceamento de carga, com o objectivo de melhor aproveitar as

capacidades de cada máquina. Conhecendo essa arquitectura, são implementadas ferramentas correspondentes aos mecanismos, as quais compõem um ambiente de grade computacional para utilizadores da linguagem Java. Uma das metas desse ambiente é permitir que utilizadores não precisem conhecer sua arquitetura e nem técnicas específicas de programação paralela para submeter trabalhos ao ambiente de alto desempenho, bastando utilizar programação concorrente multitarefa (*multithreading*) amplamente conhecida e utilizada. [Mattos, Érico 2005]

O projecto SIRENA, publicado na revista IEEE através de um artigo intitulado "*Service-Oriented Paradigms in Industrial Automation*" pretende abordar e lançar um desafio no desenvolvimento de dispositivos, aplicações e serviços, baseados nas novas tecnologias, sempre com o objectivo de aumentar e optimizar a inteligência e funcionalidades de que são providos. Faz uma descrição dos desafios enfrentados pela comunidade do "manufacturing" e das oportunidades resultantes da crescente tendência a pormenorizar e utilizar protocolos de comunicação padrão. É dada uma visão geral de como a representação dos padrões de modelos orientados a serviços, em particular quando são implementados utilizado *Web Services*, podem ajudar a atingir os objectivos inicialmente propostos pelo autor. É também descrita e avaliada a estrutura de comunicação de um modelo orientado a serviços proposta pelo SIRENA (*Service Infrastructure for Real-Time Embedded Network Applications*) que permite o uso de dispositivos industriais em redes distribuídas. A evolução e a convergência entre a computação e as tecnologias das redes distribuídas, sustentadas pelos avanços das tecnologias dos semicondutores e da transmissão de dados, limitam a revolução de como as comunicações são organizadas entre sistemas e dispositivos. Certamente, o aparecimento e criação de dispositivos cada vez mais poderosos, fáceis de interligar e conectar entre si e a redes distribuídas permitirá integrar e utilizar a inteligência da computação e das comunicações numa programação de baixo nível (ao nível do dispositivo), possibilitando, apoiado pelo protocolo Internet, uma comunicação de alto nível. Seguindo esta tendência e com a necessidade de se adaptar ao rápido desenvolvimento de arquitecturas orientadas a serviços, baseadas em padrões dos *Web Services*, o projecto SIRENA é implementado numa linguagem de alto nível que permite uma troca de dados e comunicação entre dispositivos, assim como entre dispositivos e aplicações. A abordagem feita pelo autor permite introduzir novas arquitecturas de dispositivos e permitirá de uma forma perfeita integrar arquitecturas de redes ao nível dos dispositivos, assim como ao nível de organizações. Para o autor, fica assim a promessa de prolongar o paradigma de sistemas de "*manufacturing holonic*" através da globalidade e diversidade de redes industriais. [James&Smit 2005]

Um artigo com o título "Segurança de Sistemas Ubíquos Baseada em Informações de Contexto", aborda o tema da programação ubíqua e tem como objectivo levantar questões sobre a segurança e privacidade existentes no paradigma de *Computação Ubíqua*.

A *Computação Ubíqua* sugere novas formas de interacção entre utilizador e máquina, desenvolvendo ambientes onde a implantação da tecnologia no quotidiano ocorre de modo transparente. Nesses ambientes, ditos ubíquos, é crescente a presença de mobilidade e informações como identificação, actividade, preferências e histórico de utilizadores são

monitorizadas, processadas e armazenadas por diferentes dispositivos e aplicações. [Gomes, F. ; Santos, F. ; Goularte, R. & Moreira, E. 2004]

Um outro artigo intitulado "*Ubiquitous communication systems for the electronics production industry: extending the CAMX framework*" fala sobre a necessidade de se estandardizar as relações de comunicação para a indústria electrónica. Fala do desenvolvimento de padrões IPC/CAMX (CAMX propõe mensagens padrão definidas em XML que são enviadas através de mensagens com suporte na internet em sistemas de computação distribuída orientada). Uma característica chave dos padrões de CAMX é que são extensíveis, permitindo a execução dos expansões e mantendo a compatibilidade. Em particular, os aumentos ao protocolo de comunicações de CAMX para permitir a escalabilidade das estruturas das comunicações são possíveis, com o uso das arquitecturas que envolvem a aglomeração do usuário ou de sistemas distribuídos. Da troca de dados pode ser igualmente fixado o uso de mecanismos da autenticação e da codificação. Adicionalmente, a qualidade de garantias do serviço pode ser fornecida para permitir a entrega tempo real dos dados. A descoberta dos dispositivos e de melhoria do protocolo podia igualmente ser permitida usando técnicas de Semântica *Web*. [Delamer, I.M. Lastra, J.L.M. Tuokko, R., 2004]

Outro trabalho com o título "*Information Technology Infrastructure and Solutions*" neste trabalho é discutido a praticabilidade e um Mercado de Recursos e sua execução usando plataformas de software. Sendo este caracterizado por uma flexibilidade elevada e por uma reconfiguração rápida ou adaptação da estrutura de rede de modo a enfrentar o mercado que está em constante mudança. Um Mercado de Recursos é uma ferramenta proposta no *BM_virtual enterprise architecture reference model* (BM_VEARM) como num ambiente apropriado realizar as exigências intrínsecas do modelo de A/VE, suportando a criação, da integração dinâmica, e da operação de A/VEs (*Agile / Virtual Enterprise*). O mercado dos recursos é um mercado electrónico e virtual que combina a oferta e a disputa de **Fornecedores** dos recursos, os elementos básicos que integrarão o A/VE. Os mercado electrónico existentes suportam em larga escala as características requeridas pelo mercado dos recursos. [Putnik, G. & Silva, J. 2005]

Outro trabalho com o título "*Market of Resources as a Virtual Enterprise Integration Enabler*", aborda o tema da melhoria de competitividade e de eficiência com base na flexibilidade das organizações, conseguido com recurso as tecnologias de informação e comunicação, com o modelo organizacional da A/VE como exemplo principal. Discute os requisitos para o modelo de A/VE e introduz a estrutura global e as funcionalidades do Mercado de Recursos. [Cunha, M. ; Putnik, G. ; Gunasekaran, A. & Ávila, P. 2005]

Outro trabalho intitulado "*BM_Virtual Enterprise Architecture Reference Model*", propõe arquitecturas de suporte para a operacionalidade do "Mercado de Recursos", isto é com o negócio. Para tal considera o modelo de Empresas Virtuais dinâmicas (*Virtual Enterprise model*), sua implementação e requerimentos para manter uma boa relação comercial, ou seja, condições de preço, prazos e sua reconfiguração caso haja imprevistos e manter o identidade dos participantes desconhecida. As condições atrás enunciadas devem ser asseguradas por infraestruturas

organizadas de modo a garantir um melhor desempenho que os ambientes tradicionais, como por exemplo, os motores de busca ou os mercados electrónicos. [Cunha, M. ; Putnik, G. ; Silva, J. & Santos, J. 2006]

Outro trabalho com o título "*Reconfigurable manufacturing network: An Educational Test-Bed*" introduz um conceito de um test-bed para estudar estruturas reconfiguráveis da rede de fabrico. O test-bed é composto de diversas células autónomas experimentais (EAC - *Experimental Autonomous Cells*) interligadas numa rede de produção. A estrutura da rede e a estrutura de EAC estão conceptualizadas. [Butala, P.; Sluga, A. & Putnik, G. 2006]

Um estudo intitulado "*Towards Ubiquitous Production Systems and Enterprises* (UPSE)", que aborda o tema dos Sistemas Ubíquos de Produção e Empresas Ubíquas e sua implementação crescente nos dias de hoje, apresenta o conceito base e um modelo formal para sistemas Ubíquos de produção e de computação para empresas. Refere o facto de cada vez mais se utilizar sistemas "inteligentes" de produção, que fazem suas próprias decisões locais independente do responsável pelas decisões ser um sistema centralizado. A inteligência destes sistemas é a inteligência colectiva, ou seja, é o resultando de um grande número decisões individuais, cada uma baseada em seu conhecimento local, que são guardadas e posteriormente usadas quando aparecem situações semelhantes. Refere o facto de estarmos numa época que cada pessoa tem vários computadores controlados por si, o que leva a que nestes sistemas não se possa programar individualmente cada dispositivo como é feito agora ao actualmente como no caso dos PLC's dos microcontroladores. [G. Putnik, C. Cardeira, P. Leitão, F. Restivo, J. Santos, A. Sluga, P. Butala 2007]

Passa-se agora a apresentar um caso de sucesso que envolve o recurso de Serviços *Web* e que já se encontra no mercado. Trata-se do *Web* Service da Amazon, que disponibiliza vários serviços(*Method*):

- *Amazon E-Commerce Service;*
- *Amazon Elastic Compute Cloud (Beta);*
- *Amazon Historical Pricing;*
- *Amazon Mechanical Turk (Beta);*
- *Amazon Simple Storage Service (S3);*
- *Amazon Simple Queue Service;*
- *Alexa Site Thumbnail;*
- *Alexa Top Sites;*
- *Alexa Web Information Service;*
- *Alexa Web Search .*

De seguida é explicado mais em pormenor o *Amazon E-Commerce Service*, dos serviços disponibilizados pela Amazon, este é o mais conhecido e serve para a comercialização dos seus **produtos**.
Existem muitos sites que utilizam este *Web Service* para o comércio de **produtos**. Estes sites apenas são como intermediários entre o **Cliente** final e a Amazon, ou seja, esses sites utilizam o *Web Service* da Amazon, Amazon E-Commerce Service, para aceder a Base de Dados de determinado tipo produto da Amazon, para isso disponibilizando motores de busca, o preço que é mostrado ao **Cliente** já contempla a margem de lucro do site. O **Cliente** ao fazer a encomenda, esta é processada em nome desse site que depois trata de reencaminhar a encomenda para o **Cliente** com o preço final (preço final = preço Amazon + margem).

Este serviço tem dois modos de operação:

- Search: devolve uma lista de produtos definidos por critérios (ex. Nome do autor, editora...)
- Look Up: sabemos previamente o ID do produto AISN (*Amazon Standard ID Number*). O AISN dos livros é o mesmo que o ISBN(*International Standard Book Number*), para os outros produtos é que temos de saber o AISN disponibilizado pela Amazon.

[Amazon E-Commerce Service, 2006]

CAPITULO 3

3 SOLUÇÕES ESTUDADAS

Introdução

Neste capítulo explica-se as várias arquitecturas estudadas e as suas caracteristicas.
Passa-se agora a explicar mais em pormenor cada uma das *Arquitecturas de serviços de* ***Broker*** *para empresas ágeis e virtuais* analisadas.

3.1 Arquitectura nº1

"Three levels hierarchy architecture", A selecção do Fornecedor é feita pelo Cliente.

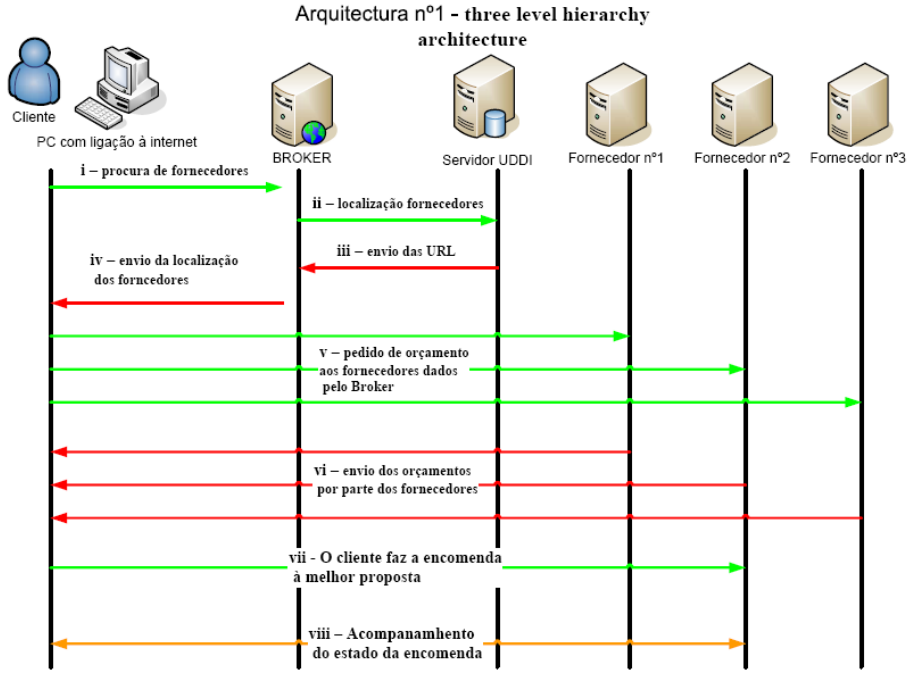

Figura 23 – ***Arquitectura*** **proposta nº 1.**

A solução proposta, ilustrada na Figura 23, prevê a existência de um servidor UDDI e de um ***Broker*** capaz de localizar e negociar de forma automática um conjunto de bens ou serviços (**Produtos**), com um ou mais **Fornecedores**, através da sua plataforma *Web*. Tanto o ***Broker*** como a Plataforma *Web* de cada **Fornecedor** implementam os Serviços *Web* propostos neste trabalho. Desta forma, um **Cliente** que pretenda adquirir um determinado **produto** ou **serviço**, poderá aceder ao site do ***Broker*** (i) e aí escolher o **produto** pretendido, as quantidades e as condições de seriação (preço ou prazo de entrega) e com base nestes requisitos, o ***Broker*** entrará

em contacto com o servidor UDDI (ii) e obterá uma lista de possíveis **Fornecedores**(iii), que em seguida serão enviados ao **Cliente** (iv) que entrará em contacto directamente com os **Fornecedores** (v) para solicitar um orçamento para o **produto** ou serviço pretendido. Depois de receber os orçamentos(vi), que podem levar semanas a chegar, o **Cliente** analisa-os e escolhe o que acha melhor e procede à encomenda do **produto** (vii). O acompanhamento do estado da encomenda é feito através de contacto directo entre o **Cliente** e o **Fornecedor** seleccionado (viii) podendo o **Fornecedor** ter este serviço ou não.

3.2 Arquitectura nº2

"Three levels hierarchy architecture", O Cliente só faz o acompanhamento da encomenda.

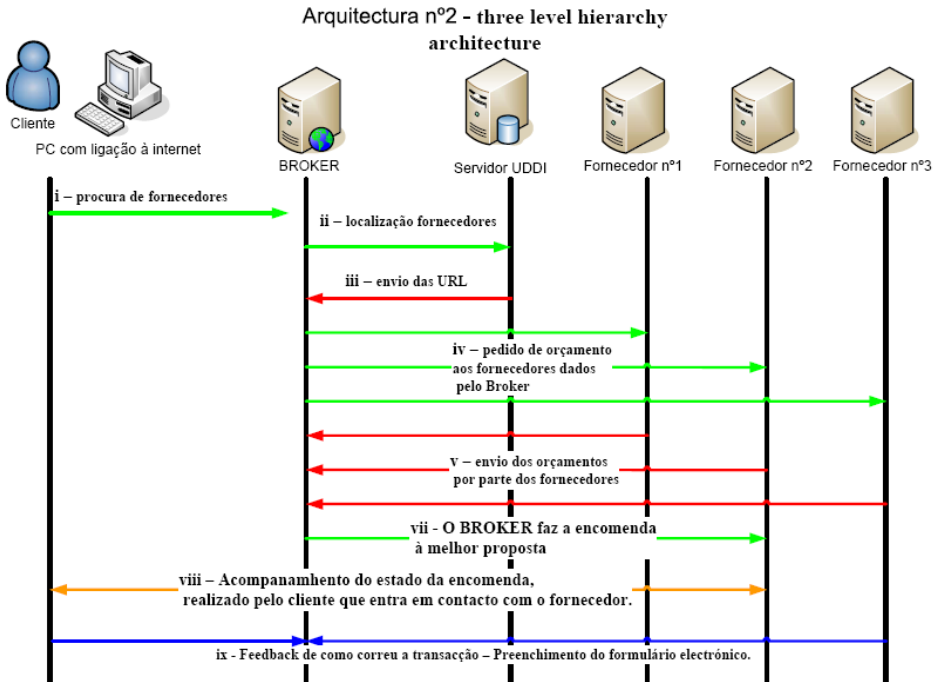

Figura 24 – *Arquitectura* proposta nº 2.

A solução proposta, ilustrada na Figura 24, prevê a existência dos mesmos intervenientes que nas arquitectura que foi apresentada anteriormente. Desta forma, um **Cliente** que pretenda adquirir um determinado **produto** ou **serviço**, poderá aceder ao site do ***Broker*** (i) e aí escolher o **produto** pretendido, as quantidades e as condições de seriação (preço ou prazo de entrega). Com base nestes requisitos, o ***Broker*** entra em contacto com o servidor UDDI (ii) e deste servidor UDDI obtem uma lista de possíveis **Fornecedores** para esse **produto**(iii). Com o conhecimento da informação necessária dos **Fornecedores**, o ***Broker*** inicia o envio a cada um dos **Fornecedores** de um pedido de cotação para o **produto** em causa(iv). Em seguida espera pelo envio dos orçamentos que será feito serviços *Web* dos **Fornecedores**(vi). À medida que os orçamentos vão

chegando o ***Broker*** vai analisa-los e coloca-los por ordem tendo em conta preço e prazos de entrega e quando tiver os orçamentos pretendidos escolhe o melhor segundo os critérios definidos pelo **Cliente** e procede então à encomenda do **produto** (vii). O ***Broker*** deverá também ser capaz de se adaptar aos imprevistos, ou seja, caso o fornecedor não consiga cumprir com o acordado, o ***Broker*** deverá ser capaz de resolver a situação. Para isso pode ter de fazer um novo pedido de cotação ou somente trocar de fornecedor tendo em conta a cotação existente. O acompanhamento do estado da encomenda é feito através de contacto directo entre o **Cliente** e o **Fornecedor** seleccionado (viii) podendo o **Fornecedor** ter ou não este serviço. Por fim, aquando a finalização da transacção quer o **Cliente**, quer o **Fornecedor** preenchem um formulário para informar o ***Broker*** de como correu a transacção. Esta informação será usada para determinar se o **Cliente** e o **Fornecedor** são de confiança, obtendo através desta avaliação possíveis melhorias financeiras pois, o **Cliente** e o **Fornecedor** vão preencher um formulário electrónico no qual vão indicar, numa escala de 1 a 5, o nível de satisfação quanto à forma como a transacção foi realizada. Com esta pontuação será definido o nível de confiança quer dos **Clientes** quer dos **Fornecedores**. Será considerado de confiança o **Cliente** que obtiver uma pontuação igual ou superior a 4 e como tal, terá a vantagem de poder pagar a encomenda só depois de a receber. Se o **Cliente** tiver pontuação máxima, terá um prazo superior de pagamento. Com este método, numa próxima encomenda, o **Cliente** e o **Fornecedor** saberão se o outro é ou não de confiança.

Algoritmo:

Para o Cliente:

Se Soma_Cliente ≤ 3 então **"Cliente não credível"** – pagamento antes de entrar em fabrico ou da entrega da encomenda.

Se Soma_Cliente ≥ 3 e ≤ 4 então **"Cliente credível"** – pagamento no acto da entrega.

Se Soma_Cliente ≥ 4 então **"Cliente de confiança"** – pagamento a 30 dias.

Se Soma_Cliente = 5 então **"Cliente de muita confiança"** – pagamento a 60 dias.

Para o Fornecedor:

Se Soma_Fornecedor ≤ 3 então **"Fornecedor não credível"**.

Se Soma_ Fornecedor ≥ 3 e ≤ 4 então **"Fornecedor credível"**.

Se Soma_ Fornecedor ≥ 4 então **"Fornecedor de confiança"**.

Se Soma_ Fornecedor = 5 então **"Fornecedor de muita confiança"**.

3.3 Arquitectura nº3

"Three levels hierarchy architecture", O Cliente dá poder de decisão ao *Broker*.

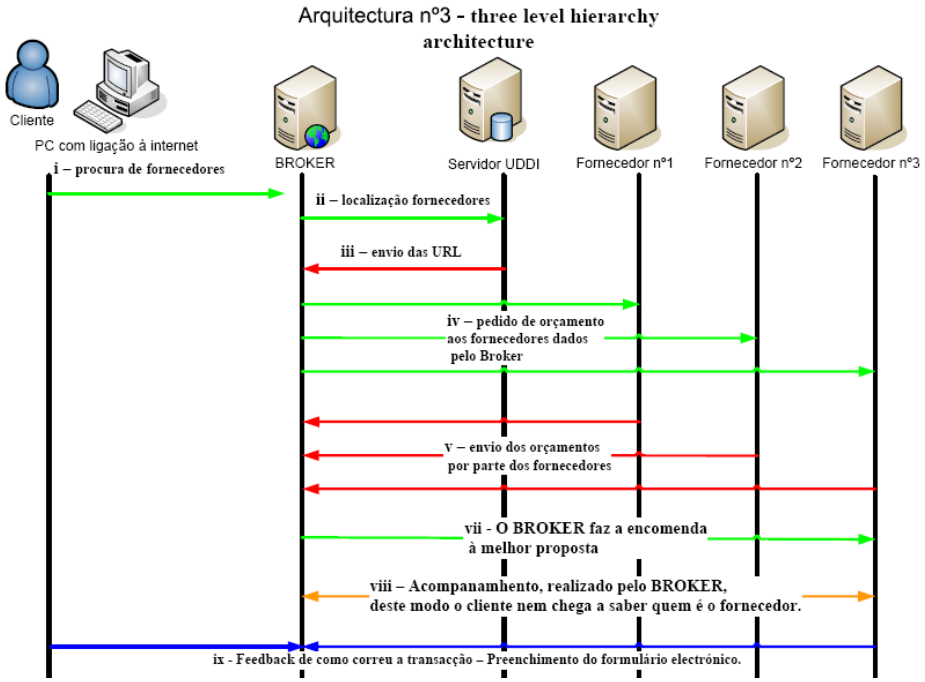

Figura 25 – *Arquitectura* proposta nº 3.

A solução proposta, ilustrada na Figura 25, prevê a existência dos mesmos intervenientes que nas *arquitecturas* n.º 1 e 2. Desta forma, um **Cliente** que pretenda adquirir um determinado **produto**, poderá aceder ao site do ***Broker*** (i) e aí escolher o **produto** pretendido, as quantidades e as condições de seriação (preço ou prazo de entrega). Com base nestes requisitos, o ***Broker*** entrará em contacto com o servidor UDDI (ii) e obterão uma lista de possíveis **Fornecedores**(iii), em seguida entrará em contacto directamente com os **Fornecedores** seleccionados para solicitar um orçamento para o **produto** pretendido (iv). É o ***Broker*** que vai analisando e colocando cada orçamento por ordem, tendo em conta preços e prazos de entrega e, quando tiver os orçamentos todos escolhe o melhor tendo em conta os critérios definidos pelo **Cliente** e procede à encomenda do **produto**(v), o ***Broker*** deverá também ser capaz de se adaptar aos imprevistos, ou seja, caso o **Fornecedor** não consiga cumprir com o acordado, o ***Broker*** deverá ser capaz de resolver a situação. Para isso pode ter de fazer um novo pedido de cotação ou somente trocar de **Fornecedor** tendo em conta a cotação existente. O acompanhamento do estado da encomenda é feito pelo ***Broker*** (vii). Por fim, aquando a finalização da transacção quer o **Cliente** quer o **Fornecedor** preenchem um formulário para informar o ***Broker*** de como correu a transacção, esta

informação será usada para classificar os **Cliente** e os **Fornecedores** segundo o critério apresentado anteriormente.

3.4 Arquitectura nº4

"Four levels hierarchy architecture", Existência de um Regulador e é o Cliente a seleccionar o Fornecedor.

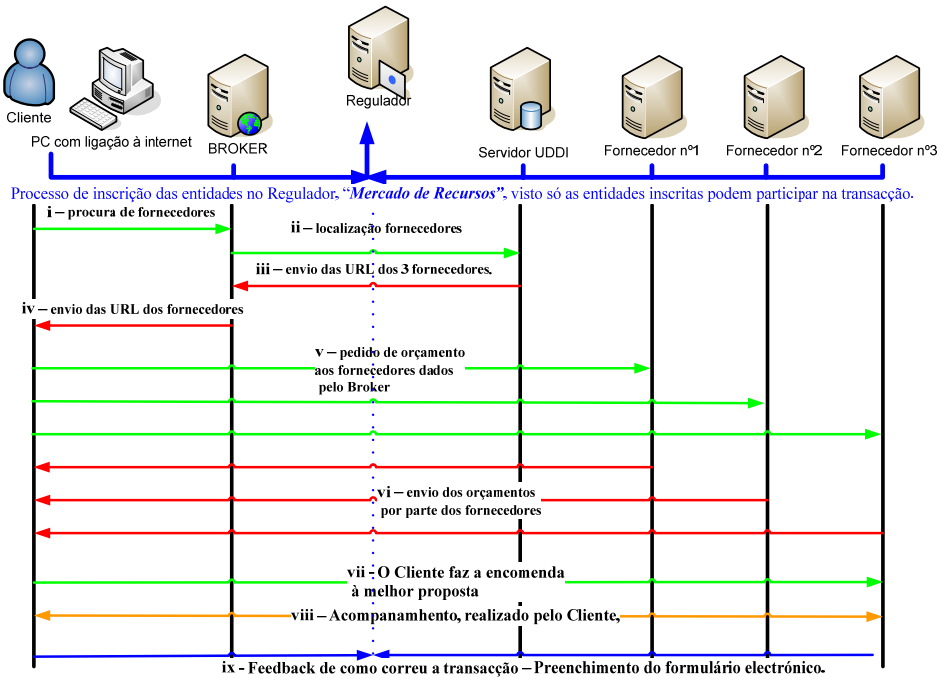

Figura 26 – *Arquitectura* proposta nº 4.

A solução proposta, ilustrada na Figura 26, prevê a existência de um Regulador, que é a entidade que monitoriza a transacção e onde todos os intervenientes têm de estar inscritos para poderem entrar na transacção, um servidor UDDI e um ***Broker*** capaz de localizar e negociar de forma automática um conjunto de bens ou serviços (**Produtos**), com um ou mais **Fornecedores**, através da sua plataforma *Web*. Tanto o ***Broker*** como a Plataforma *Web* de cada **Fornecedor** implementam os Serviços *Web* propostos neste trabalho. Desta forma, um **Cliente** que pretenda adquirir um determinado **produto**, poderá aceder ao site do ***Broker*** (i) e aí escolher o **produto** pretendido, as quantidades e as condições de seriação (preço ou prazo de entrega). Com base nestes requisitos, o ***Broker*** entrará em contacto com o servidor UDDI (ii) e obterá uma lista de possíveis **Fornecedores** que têm de estar inscritos no *Regulador* (iii). O ***Broker*** envia a lista dos **Fornecedores** para o **Cliente**(iv) que entra em contacto com os **Fornecedores** para pedir a cotação para o **produto** em causa(v), depois espera pelo envio dos orçamentos que será feito serviços *Web* dos **Fornecedores**(vi). À medida que os orçamentos vão chegando o **Cliente** vai

analisa-los e colocá-los por ordem tendo em conta preço e prazos de entrega e quando tiver todos os orçamentos escolhe o melhor e procede à encomenda do **produto** (vii). O acompanhamento do estado da encomenda é feito através de contacto directo entre o **Cliente** e o **Fornecedor** seleccionado (viii) podendo este demorar um pouco mais, visto depender do Homem. Por fim, aquando a finalização da transacção quer o **Cliente**, quer **Fornecedor** preenchem um formulário para informar o *Regulador* de como correu a transacção. A informação contida nesses formulários irá ser usada para actualizar a informação relativa aos **Cliente**s e aos **Fornecedores**, para assim se poder actualizar a sua classificação de acordo com o que já se apresentou nas arquitecturas nº 2 e 3.

3.5 Arquitectura nº5

"Four levels hierarchy architecture", Existência do Regulador e é o *Broker* a tratar da transacção.

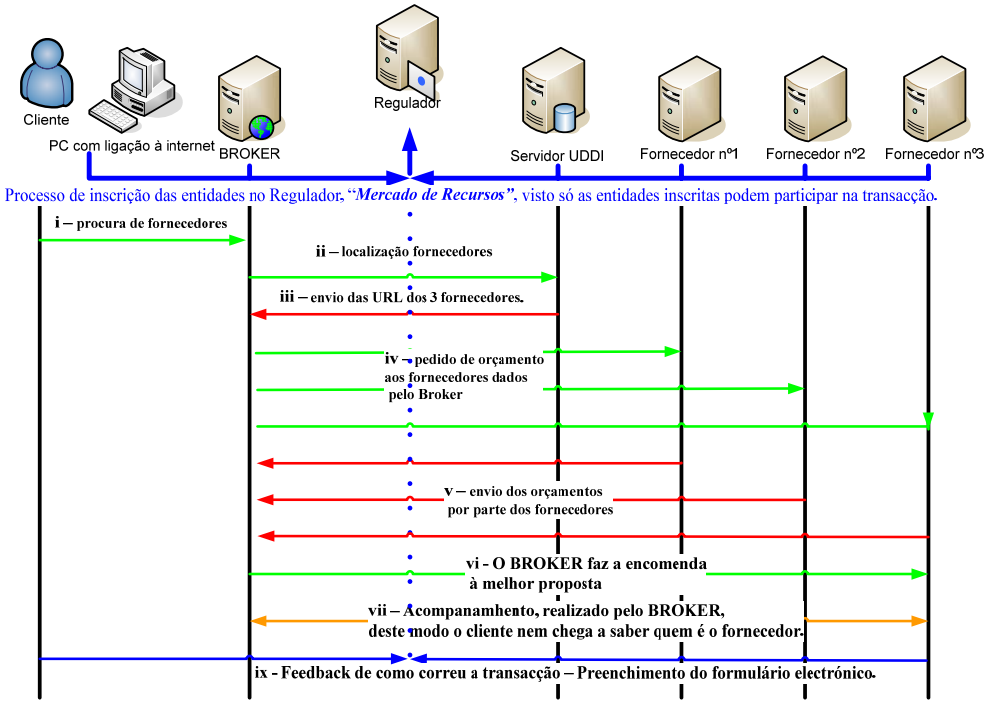

Figura 27 – *Arquitectura* proposta nº 5.

A solução proposta, ilustrada na Figura 27,prevê a existência dos mesmos intervenientes que na *arquitectura* n.º 4 (Figura 26).

Assim sendo, um **Cliente** para adquirir um determinado **produto**, terá de aceder ao site do ***Broker*** (i) e aí escolher o **produto** pretendido, as quantidades e as condições de seriação (preço ou prazo de entrega). Com base nestes requisitos, o ***Broker*** entra em contacto com o servidor UDDI (ii) para obter uma lista de possíveis **Fornecedores**, todos eles inscritos no na Base de Dados do

Regulador (iii). Com esta lista dos **Fornecedores** o ***Broker*** envia aos **Fornecedores** um pedido de cotação para o **produto** em causa(iv) e depois espera pelo envio dos orçamentos que será feito serviços *Web* dos **Fornecedores**(v). É o ***Broker*** que vai analisando e colocando cada orçamento por ordem, tendo em conta preços e prazos de entrega e, quando tiver os orçamentos todos escolhe o melhor e procede à encomenda do **produto** *(vi)*. O ***Broker*** deverá também ser capaz de se adaptar aos imprevistos e adaptar-se rapidamente às novas situações de modo a garantir o bom funcionamento da transacção. Ou seja, caso o **Fornecedor** não consiga cumprir com o acordado, o ***Broker*** deverá ser capaz de resolver a situação, tendo por isso de fazer um novo pedido de cotação a vários **Fornecedores** (repetindo por isso os pontos ii, iii iv, v e vi) ou somente trocar de **Fornecedor** tendo em conta a cotação existente. O acompanhamento do estado da encomenda é feito pelo ***Broker*** (vii).

Por fim, aquando a finalização da transacção quer o **Cliente***,* quer **Fornecedor** preenchem um formulário para informar o *Regulador* de como correu a transacção. A informação contida nesses formulários irá ser usada para actualizar a informação relativa aos **Cliente***s* e aos **Fornecedor*es***, para assim se poder actualizar a sua classificação de acordo com o que já se apresentou nas arquitecturas nº 2 e 3 e 4.

CAPITULO 4

4 ESTUDO COMPARATIVO ENTRE ARQUITECTURAS ESTUDADAS

4.1 Arquitectura nº1 – ”Three level hierarchy architecture”, Cliente é quem tem o poder de decisão.

4.1.1 Vantagens

No caso desta *arquitectura* a principal vantagem é a de envolver poucos intervenientes o que a torna menos complexa. É o **Cliente** que decide tudo, desde dos **Fornecedores** que pretende contactar, ao orçamento que deseja até à realização da encomenda. A única função do ***Broker*** é a de informar o **Cliente** acerca dos potenciais **Fornecedores**, indicados pelos servidores UDDI.
Existe pouca dependência relativamente às máquinas (***Broker***) pois, o **Cliente** é o maior interveniente e está menos sujeito a ser prejudicado por falhas das máquinas. Pelo facto do ***Broker*** fazer somente a pesquisa sobre os **Fornecedores** nos servidores UDDI, a sua complexidade é mínima e não precisa de ter uma programação muito elaborada.

- **Complexidade da arquitectura:**

Esta arquitectura tem a vantagem que envolve **poucos intervenientes**. É o C*liente* que decide tudo desde quais os **Fornecedores** que pretende contactar e qual o orçamento que pretende até à encomenda. O ***Broker*** só tem de informar quem são os potenciais **Fornecedores**, que são indicados pelos servidores UDDI.

- **Pouca dependência e complexidade (*Broker*) das máquinas:**

Está menos dependente de máquinas ou seja das falhas das mesmas.

- **Pouca complexidade do *Broker*:**

Como o ***Broker*** só faz a pesquisa sobre os **Fornecedores** nos servidores UDDI não precisa de ter uma programação muito complexa.

4.1.2 Desvantagens

O factor tempo é uma das desvantagens desta arquitectura, o facto do **Cliente** ter um papel fundamental faz com que as diferentes tarefas possam ser realizadas lentamente. Isto porque o **Cliente** é que faz todos os contactos para pedir orçamentos, estando dependente da rapidez do **Fornecedor**. É também o **Cliente** que analisa, decide qual é a melhor proposta e encarrega-se de fazer a encomenda.
Outro inconveniente desta arquitectura reside no acompanhamento da encomenda. Visto a encomenda ser realizada directamente entre o **Cliente** e os serviços *Web* do **Fornecedor**, não é garantido o acompanhamento da encomenda ao longo do seu percurso até chegar às mãos do **Cliente**. Ou seja, depois de ser enviada pelo **Fornecedor** para as transportadores pode perder-se o "contacto" com a mercadoria. O facto de haver contacto directo entre os **Cliente** e **Fornecedores** faz com que não haja garantias quanto à confidencialidade dos dados dos diversos intervenientes. A Segurança e Qualidade da transacção também são factores a ter em conta pois são aspectos que não são garantidos. Quer o **Cliente**, quer os **Fornecedores** podem não ser de confiança e assim sendo, o negócio não corre como planeado. O mesmo se aplica relativamente ao pagamento da encomenda visto não haver forma de garantir que o **Cliente** e o **Fornecedor** cumprem com o combinado no acto da encomenda.

- **Tempo de demora elevado:**

Como nesta arquitectura o **Cliente** tem papel fundamental faz com que ela possa ser muito demorada. Isto porque o **Cliente** é que faz todos os contactos para pedir orçamentos, quem trata de os seleccionar e escolher o melhor.

- **Acompanhamento da encomenda:**

Como é feito directamente entre **Cliente** e **Fornecedor** não é garantido que se sabe o estado da encomenda, ou seja, depois de esta ser enviada para as transportadores pode-se perder o "contacto" com a mercadoria.

- **Segurança/Qualidade da transacção:**

Não é garantida a Segurança/Qualidade da transacção, ou seja, quer o C*liente* como os **Fornecedores** podem não ser de confiança, não é garantida a sua fidegnidade, e assim sendo o negócio pode não correr como pretendido.

- **Confidencialidade:**

Há contacto directo entre os **Cliente** e **Fornecedores**, ou seja, sabe-se quem são as entidades envolvidas.

- **Pagamento:**

Não é possível garantir que o **Cliente** e o **Fornecedor** cumprem com o combinado.

4.2 Arquitectura nº2 – ”Three level hierarchy architecture”, Cliente só trata do acompanhamento do estado da transacção.

4.2.1 Vantagens

A grande vantagem desta *arquitectura* é o tempo de demora reduzido, visto ser o ***Broker*** a tratar de tudo a partir do momento em que o **Cliente** solicita aquela quantidade e prazo de entrega para determinado **produto**. Depois do pedido do **Cliente**, o ***Broker*** é que trata desde a pesquisa de **Fornecedores**, os pedidos de orçamento, selecção do melhor orçamento e formalização da encomenda.

- **Tempo de demora reduzido:**

Como nesta arquitectura é o ***Broker*** que trata de tudo, ou seja, depois do pedido de contacto do **Cliente** o ***Broker*** é que trata desde a pesquisa de **Fornecedores**, dos pedidos de orçamento até a selecção do melhor orçamento.

4.2.2 Desvantagens

Uma das desvantagens desta arquitectura é o facto do acompanhamento do estado da encomenda ser feito directamente pelo **Cliente** ao **Fornecedor**, este pedido de informação pode não ser o mais correcto visto poder demorar algum tempo a responder.
A Segurança e Qualidade da transacção também não são garantidos. Visto que quer o **Cliente** quer os **Fornecedores** podem não ser de confiança, não é garantida a sua fidedignidade, e assim sendo o negócio pode não correr como planeado.
Contrariamente à arquitectura anterior, existe aqui uma maior dependência do **Cliente** relativamente às máquinas. Como ainda não está garantida a 100% a fiabilidade das máquinas o **Cliente** pode estar sujeito a falhas, neste caso do ***Broker***, do servidor UDDI mas também dos servidores dos **Fornecedores**.
Quanto à confidencialidade, esta também não está garantida visto haver contacto directo entre o **Cliente** e o **Fornecedor** escolhido, portanto sabe-se quem são as entidades envolvidas directamente na transacção.
Existe uma grande complexidade na programação do ***Broker***, porque é o ***Broker*** a tratar praticamente de tudo, à excepção do acompanhamento da encomenda que é feita pelo próprio **Cliente** directamente com o **Fornecedor**.
No que se refere ao pagamento, como as entidades que entram na transacção podem não ser de confiança, não há forma directa de garantir que o **Cliente** paga o que encomendou e que o

Fornecedor, fornece o que foi combinado. No entanto é feita uma avaliação dos intervenientes na transacção. Existe, no início do processo a criação de algoritmos de controle quer dos **Clientes** quer dos **Fornecedores**. Tanto os **Clientes** como os **Fornecedores** têm de se inscrever no ***Broker*** e recebem uma password de acesso à sua conta. Depois da transacção se realizar, o **Cliente** e o **Fornecedor** terão de preencher um formulário electrónico no qual vão indicar, numa escala de 1 a 5, o nível de satisfação quanto à forma como a transacção foi realizada, sendo depois estes níveis de satisfação usados para calcular se o **Cliente** e **Fornecedor** são de confiança, como se explicou no Capítulo acima.

- **Acompanhamento da encomenda:**

Como é feito directamente entre **Cliente** e **Fornecedor** não é garantido que se sabe o estado da encomenda, ou seja, depois de ser enviada pelo **Fornecedor** para as transportadores pode-se perder o "contacto" com a mercadoria.

- **Segurança/Qualidade da transacção:**

Não é garantida a Segurança/Qualidade da transacção, ou seja, quer o C*liente* como os **Fornecedores** podem não ser de confiança, não é garantida a sua fidegnidade, e assim sendo o negócio pode não correr como devia.

- **Dependência das máquinas:**

Como ainda não está garantida a 100% a fiabilidade das máquinas, neste caso do ***Broker***, do servidor UDDI mas também dos servidores dos **Fornecedores**. Isto porque como é o ***Broker*** que trata da parte da negociação temos de este não tem meios para fazer "marralhar" os preços.

- **Confidencialidade:**

Há contacto directo entre os **Cliente** e **Fornecedor** escolhido, ou seja, sabe-se quem são as entidades envolvidas directamente na transacção.

- **Grande complexidade do *Broker*:**

Como o ***Broker*** trata de todo a transacção, só não faz a parte do acompanhamento da encomenda, que é feito pelo **Cliente** directamente com o **Fornecedor**, já precisa de ser mais complexo em termos de programação visto o ***Broker*** deverá ser capaz de se adaptar aos imprevistos, ou seja, caso o **Fornecedor** não consiga cumprir com o acordado, o ***Broker*** deverá ser capaz de resolver a situação. Para isso pode ter de fazer um novo pedido de cotação ou somente trocar de **Fornecedor** tendo em conta a cotação existente.

- **Pagamento:**

Como as entidades que entram na transacção podem não ser de confiança, não há maneira directa de garantir que o **Cliente** paga o que encomendou e que o **Fornecedor**, fornece o que foi combinado. Para tentar resolver essa situação criam-se os algoritmos de controlo que se falou no Capítulo anterior, para que assim já se pudesse ter uma ideia da ideonidade dos participantes na transacção.

4.3 Arquitectura nº3 – ”three level hierarchy architecture”, *Broker* trata da transacção.

4.3.1 Vantagens

Como na *arquitectura* nº2, aqui também é o ***Broker*** que trata de tudo, ou seja, depois do pedido do **produto** ou **serviço** feito pelo **Cliente**, o ***Broker*** é que trata desde a pesquisa de **Fornecedores**, aos pedidos de orçamento até a selecção do melhor orçamento. Isto torna o tempo de demora mais reduzido.
Contrariamente às outras *arquitectura*, pelo facto do **Cliente** não entrar em contacto directo com o **Fornecedor**, sendo todo processo tratado pelo ***Broker***, existe confidencialidade dos dados do **Cliente** mas também do **Fornecedor**.
O acompanhamento da encomenda está assegurado porque ele é feito pelo ***Broker*** e pelos Serviços *Web* disponibilizados pelo **Fornecedor**. São portanto garantidas as informações sobre o estado da encomenda até chegar às mãos do **Cliente**.

- **Tempo de demora reduzido:**

Como nesta arquitectura é o ***Broker*** que trata de tudo, ou seja, depois do pedido de contacto do **Cliente** o ***Broker*** é que trata desde a pesquisa de **Fornecedores**, dos pedidos de orçamento até a selecção do melhor orçamento.

- **Confidencialidade:**

O **Cliente** não entra em contacto directo com o **Fornecedor**, todo o processo é tratado pelo ***Broker***.

- **Acompanhamento da encomenda:**

Como é feito pelo ***Broker*** e **Fornecedor** é garantido que se sabe o estado da encomenda.

4.3.2 Desvantagens

Como já se demonstrou nas *arquitecturas* anteriores a Segurança e Qualidade da transacção não são garantidas. O **Cliente** e os **Fornecedores** podem não ser de confiança, não é garantida a sua fidedignidade, e assim sendo o negócio pode não correr como estava definido inicialmente.
Como na arquitectura nº 2, não existem garantias a 100% quanto à fiabilidade das máquinas. O Processo de compra é muito dependente da actuação das máquinas o que pode ser prejudicial para o **Cliente** caso haja falhas por parte das máquinas (do ***Broker***, do servidor UDDI mas também dos serviços *Web* dos **Fornecedores**). Aqui também existe o inconveniente de como é o ***Broker*** a tratar da parte da escolha de orçamento e da encomenda, os meios para negociar os

preços poderem não ser os mais eficazes. Há também uma grande complexidade do ***Broker*** em termos de programação visto ser o ***Broker*** a tratar de toda a transacção, desde a pesquisa de **Fornecedores**, ao pedido de orçamentos, escolha do melhor orçamento até ao acompanhamento da encomenda.

No que se refere ao pagamento da encomenda, com esta *arquitectura* não há maneira directa de garantir que o **Cliente** paga o que encomendou e que o **Fornecedor**, fornece o que foi combinado, tenta-se resolver este problema usando a solução apresentada na arquitectura nº2.

- **Segurança/Qualidade da transacção:**

Não é garantida a Segurança/Qualidade da transacção, ou seja, quer o **Cliente** como os **Fornecedores** podem não ser de confiança, não é garantida a sua fidegnidade, e assim sendo o negócio não corre como devia.

- **Dependência das máquinas:**

Como ainda não está garantida a 100% a fiabilidade das máquinas, neste caso do ***Broker***, do servidor UDDI mas também dos servidores dos **Fornecedores**. Isto porque como é o ***Broker*** que trata da parte da negociação temos de este não tem meios para fazer "marralhar" os preços.

- **Grande complexidade do *Broker*:**

Como o ***Broker*** trata de toda a transacção, desde a pesquisa de **Fornecedores**, do pedido de orçamentos, escolha do melhor orçamento até ao acompanhamento da encomenda, ainda de ter "inteligência" para ser capaz de decidir o que fazer caso o **Fornecedor** escolhido não seja capaz de fornecer dentro do prazo estipulado, nestas situações o ***Broker*** pode seleccionar o **Fornecedor** que ficou em segundo lugar aquando da escolha, isto se as condições se mantiverem e o prazo o permitir, senão terá de fazer nova pesquisa e nova selecção de **Fornecedor** tendo em conta os novos critérios(data e quantidade).

Por estas razões precisa de ser um ***Broker*** mais complexo em termos de programação.

- **Pagamento:**

Como as entidades que entram na transacção podem não ser de confiança, não há maneira directa de garantir que o **Cliente** paga o que encomendou e que o **Fornecedor**, fornece o que foi combinado. Para isso criam-se algoritmos de controlo quer dos **Clientes** mas também dos **Fornecedores**, por isso quer os **Clientes** quer os **Fornecedores** têm de se inscrever no ***Broker*** (recebem uma password de acesso à sua conta). Depois da transacção se realizar, o **Cliente** e o **Fornecedor** terão de preencher um formulário electrónico no qual indicaram numa escala de 1 a 5 o quanto estão contentes com a transacção. Com esta pontuação o **Cliente** se obtiver uma pontuação igual ou superior a 4 será de confiança, e como tal terá a vantagem de poder pagar a encomenda só depois de a receber. Se o **Cliente** tiver pontuação máxima, terá um prazo superior de pagamento. Com este método o **Cliente** e o **Fornecedor** sabem se o outro é ou não de confiança.

4.4 Arquitectura nº4 – ”Four level hierarchy architecture”, Regulador supervisiona a transacção, indica os Fornecedores de confiança.

4.4.1 Vantagens

A grande vantagem desta *arquitectura* é a qualidade e ter a garantia do pagamento da transacção, isto porque para poder participar na transacção os intervenientes têm de estar inscritos no Regulador tendo por isso de fornecer os seus dados. Tem ainda a vantagem de envolver poucos intervenientes o que a torna menos complexa. É o **Cliente** que decide tudo, desde dos **Fornecedores** que pretende contactar, ao orçamento que deseja até à realização da encomenda. A única função do ***Broker*** é a de informar o **Cliente** acerca dos potenciais **Fornecedores**, indicados pelos servidores UDDI.

Existe pouca dependência relativamente às máquinas (***Broker***) pois, o **Cliente** é o maior interveniente e está menos sujeito a ser prejudicado por falhas das máquinas. Pelo facto do ***Broker*** fazer somente a pesquisa sobre os **Fornecedores** nos servidores UDDI, a sua complexidade é mínima e não precisa de ter uma programação muito elaborada.

- **Tempo de demora reduzido:**

Como nesta arquitectura é o ***Broker*** que trata de tudo, ou seja, depois do pedido de contacto do **Cliente** o ***Broker*** é que trata desde a pesquisa de **Fornecedores**, dos pedidos de orçamento até a selecção do melhor orçamento.

- **Confidencialidade:**

O **Cliente** não entra em contacto directo com o **Fornecedor**, todo o processo é tratado pelo ***Broker***.

- **Acompanhamento da encomenda:**

Como é feito pelo ***Broker*** e **Fornecedor** é garantido que se sabe o estado da encomenda.

- **Segurança/Qualidade da transacção:**

É garantida a segurança/qualidade do **Fornecedor** visto existir uma entidade que regula toda a transacção, *Regulador de Mercado.*

- **Pagamento:**

Como todos os intervenientes estão inscritos e são obrigados a preencher formulários que “garantem” a sua ideonidade, o pagamento feito pelo **Cliente** é feito depois de receber a encomenda.

4.4.2 Desvantagens

O factor tempo é uma das desvantagens desta arquitectura, o facto do **Cliente** ter um papel fundamental faz com que as diferentes tarefas possam ser realizadas lentamente. Isto porque é o **Cliente** que solicita os orçamentos, analisa-os, decide qual é a melhor proposta e encarrega-se de fazer a encomenda.

- **Dependência das máquinas:**

Como ainda não está garantida a 100% a fiabilidade das máquinas, neste caso do ***Broker***, do servidor UDDI mas também dos servidores dos **Fornecedores**. Isto porque como é o ***Broker*** que trata da parte da negociação temos de este não tem meios para fazer "marralhar" os preços.

- **Grande complexidade do *Broker*:**

Como o ***Broker*** trata de toda a transacção, desde a pesquisa de **Fornecedores**, do pedido de orçamentos, escolha do melhor orçamento até ao acompanhamento da encomenda, ainda de ter "inteligência" para ser capaz de decidir o que fazer caso o **Fornecedor** escolhido não seja capaz de fornecer dentro do prazo estipulado, nestas situações o ***Broker*** pode seleccionar o **Fornecedor** que ficou em segundo lugar aquando da escolha, isto se as condições se mantiverem e o prazo o permitir, senão terá de fazer nova pesquisa e nova selecção de **Fornecedor** tendo em conta os novos critérios (data e quantidade).

Por estas razões precisa de ser um ***Broker*** mais complexo em termos de programação.

- **Número de intervenientes:**

Existência de mais um elemento, *Regulador*, que faz com que a complexidade da arquitectura se torne maior.

4.5 Arquitectura nº5 –"Four level hierarchy architecture", Todos os intervenientes são membros do Regulador da transacção.

4.5.1 Vantagens

A grande vantagem desta *arquitectura* é o tempo de demora reduzido, visto ser o ***Broker*** a tratar de tudo a partir do momento em que o **Cliente** solicita a pesquisa de **Fornecedores**. Depois do pedido de contacto do **Cliente** o ***Broker*** é que trata desde a pesquisa de **Fornecedores**, aos pedidos de orçamento, selecção do melhor orçamento até à formalização da encomenda.

A qualidade e o pagamento da transacção são garantidas porque para poder participar na transacção os intervenientes têm de estar inscritos no *Regulador* tendo por isso de fornecer os seus dados.

Outra vantagem é o facto dos intervenientes na transacção não serem divulgados, havendo assim confidencialidade.

- **Tempo de demora reduzido:**

Como nesta arquitectura é o ***Broker*** que trata de tudo, ou seja, depois do pedido de contacto do **Cliente** o ***Broker*** é que trata desde a pesquisa de **Fornecedores**, dos pedidos de orçamento até a selecção do melhor orçamento.

- **Confidencialidade**:

O **Cliente** não entra em contacto directo com o **Fornecedor**, todo o processo é tratado pelo ***Broker***.

- **Acompanhamento da encomenda:**

Como é feito pelo ***Broker*** é garantido que se sabe o estado da encomenda.

- **Segurança/Qualidade da transacção:**

É garantida a segurança/qualidade de toda a transacção, isto porque todos os intervenientes estão são membros do *Regulador*.

- **Pagamento:**

Como todos os intervenientes estão inscritos e são obrigados a preencher formulários que "garantem" a sua ideonidade, o pagamento feito pelo **Cliente** é feito depois de receber a encomenda.

4.5.2 Desvantagens

Ao contrario da arquitectura anterior, existe aqui uma dependência do **Cliente** relativamente às máquinas. Como ainda não está garantida a 100% a fiabilidade das máquinas, a demora da transacção pode estar em causa por causa das falhas das máquinas, como é o caso do *Regulador*, do ***Broker***, do servidor UDDI mas também dos serviços *Web* dos **Fornecedores**.

Existe também o facto de, como é o ***Broker*** a tratar da parte da escolha de orçamento e da encomenda, os meios para negociar os preços poderem não ser os mais eficazes.

- **Dependência das máquinas:**

Como ainda não está garantida a 100% a fiabilidade das máquinas, neste caso do ***Broker***, do servidor UDDI mas também dos servidores dos **Fornecedores**. Isto porque como é o ***Broker*** que trata da parte da negociação temos de este não tem meios para fazer "marralhar" as condições da proposta.

- **Grande complexidade do *Broker*:**

Como é o ***Broker*** o responsável por toda a transacção, desde a pesquisa de **Fornecedores**, do pedido de orçamentos e a selecção do melhor orçamento, vai ter ainda de ter "inteligência" para ser capaz de decidir o que fazer caso o **Fornecedor** escolhido não seja capaz de fornecer dentro do prazo estipulado. Nestas situações o ***Broker*** pode seleccionar o **Fornecedor** que ficou em segundo lugar aquando da escolha, isto se as condições se mantiverem e o prazo o permitir, senão terá de fazer nova pesquisa e nova selecção de **Fornecedor** tendo em conta os novos critérios(data e quantidade).

- **Número de intervenientes:**

Existência de mais um elemento, *Regulador*, que faz com que a complexidade da arquitectura se torne maior.

4.6 Tabela Comparativa das Arquitecturas:

Tabela 4 – Tabela comparativa das arquitecturas estudadas.

	Arquitectura nº1	**Arquitectura nº2**	**Arquitectura nº3**	**Arquitectura nº4**	**Arquitectura nº5**
Tempo de demora da transacção	Elevado	Reduzido	Reduzido	Reduzido	Reduzido
Confidencialidade	Não existe	Parcial	Garantida	Garantida	Garantida
Dependência das máquinas	Reduzida	Elevada	Elevada	Elevada	Elevada
Segurança / Qualidade da transacção	Não é garantida	Não é garantida	Não é garantida	Parcial	Garantida
Acompanhamento do estado da encomenda	Ineficaz	Ineficaz	Eficaz	Eficaz	Eficaz
Complexidade do Broker	Reduzida	Elevada	Elevada	Elevada	Elevada
Pagamento	Antes de receber a encomenda	Consoante a pontuação do **Cliente**	Consoante a pontuação do **Cliente**	Assegurado, Facilidade de pagamento	Assegurado, Facilidade de pagamento

Observando a Tabela 4 e comparando as cinco arquitecturas apresentadas, pode-se concluir que a melhor arquitectura é a quinta (5º) arquitectura, visto a transacção ser rápida, por ser realizada só por máquinas que são membros do *Regulador*, mais segura / qualidade e o **Cliente** tem maior facilidade de pagamento. Porém esta arquitectura tem um ponto menos positivo, que é a complexidade que o ***Broker*** terá de ter, visto ter de fazer toda transacção, ou seja, trata de descobrir os **Fornecedores**, pedir orçamentos, escolher o melhor, deverá ter capacidade de se adaptar aos imprevistos, ser ágil e flexível.

CAPITULO 5

5 SOLUÇÃO PROPOSTA

5.1 Introdução

A solução proposta, tal como referido no capítulo 1.5, recorre à utilização de um ***Broker*** para descobrir os **Fornecedores** capazes de fornecerem um produto ou serviço a um **Cliente**, ao melhor preço e/ou o mais rapidamente possível, de forma simples, autónoma, rápida e automática. Esta solução pretende também que toda a informação disponibilizada esteja sempre o mais actualizada possível e seja fidedigna. Assim, na solução proposta, sempre que um **Cliente** faz um pedido no nosso sítio, este pedido irá ser analisado pelo Broker associado, que irá consultar os *Web Services* disponibilizados pelos **Fornecedores** que retornarão a resposta ao pedido feito. Desta forma, toda a informação disponibilizada ao **Cliente** tem como fonte os próprios **Fornecedores,** estando assim o mais actualizada e sendo o mais fidedigna possível, excepto na situação de haverem falhas nos próprios **Fornecedores**.

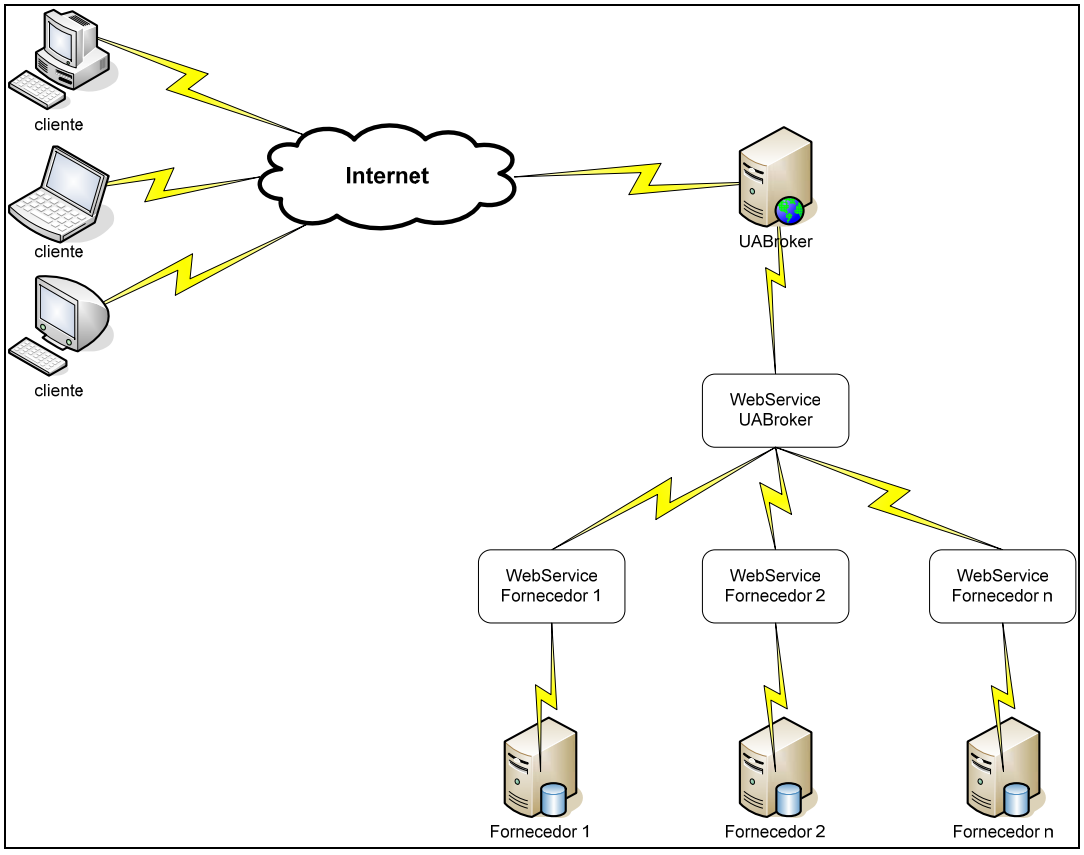

Figura 28 – Diagrama da aplicação

5.2 Tecnologias de Suporte Utilizadas

A tecnologia de suporte utilizada para a realização deste trabalho foi *Web Services*, isto depois de analisadas as suas vantagens e desvantagens relativamente a outras tecnologias de suporte existentes.

O uso de *middlewares* baseados em protocolos de comunicação apesar de ter tido uma grande aceitação em redes locais e partilhadas, onde as restrições de segurança são adoptadas de acordo com os *middlewares* utilizados nessas redes, quando adoptados nas redes distribuídas locais e partilhadas podem não conseguir ultrapassar os sistemas de segurança (*firewalls*) das redes que partilham dados na Internet, daí a sua não utilização neste trabalho.

Para a realização da solução proposta neste trabalho recorreu-se então ao uso de tecnologias de suporte dos *Web Services*, visto o objectivo deste trabalho ser descobrir e contactar os vários **Fornecedores** para o fornecimento de um determinado **Produto** utilizando a Internet como infra-estrutura de comunicação. Por isso o meio de comunicação no qual se iriam transmitir os dados ou informações – a Internet e os seus sistemas de segurança – foi um parâmetro importante na escolha do uso de *Web Services*.

Os *Web Services* são a mais recente evolução nos padrões de desenvolvimento de aplicações distribuídas permitindo que aplicações cooperem facilmente e compartilhem informações e dados entre si. Estes são serviços distribuídos que processam mensagens SOAP codificadas em XML, enviadas através de HTTP, e que são descritas através da *Web Services Description Language* (WSDL). O objectivo dos *Web Services* é fornecer um determinado serviço a outras aplicações ou mesmo a outros Serviços *Web*. Pode ser traduzido numa simples validação ou numa informação mais detalhada como a que pode ser encontrada na solução proposta (prazo, preço, identificação do **Fornecedor***)*. Uma das grandes vantagens da utilização dos *Web Services* é o facto de serem simples e fáceis de implementar, transparentes e isentos de quaisquer custos e pagamentos de patentes, pois utilizam protocolos padrão da Internet (SOAP, HTTP, WSDL), um formato padrão de troca de informações (XML) e um directório de registo e localização (UDDI). São estes os factores que fornecem pré-requisitos importantes para a larga aceitação dos *Web Services* face a outras arquitecturas, tais como o CORBA e o DCE, que se encontram ligadas a protocolos específicos e que precisam de ser conhecidos pelas camadas que pretendem comunicar entre si.

Outra grande vantagem da utilização desta tecnologia de suporte é o facto da sua integração ser relativamente simples, bem como a implementação de novos serviços ou novas funções. No que diz respeito ao aspecto da segurança verifica-se que a comunicação está mais segura, uma vez que apenas há transmissão de informação estritamente necessária.

Apesar dos *Web Services* serem mais utilizados quando se pretende disponibilizar ou trocar dados na Internet, uma vez que permitem uma comunicação assíncrona eliminando os problemas relacionados com *firewalls* quando utilizados como tecnologia de suporte, em sistemas distribuídos de redes locais e/ou partilhadas, não apresentam um desempenho idêntico ao do CORBA e ao do

DCE. Além de que o tratamento dos documentos XML é bastante mais complexo quando comparado com os dos objectos distribuídos utilizados no CORBA ou DCE.
Os protocolos de comunicação utilizados numa transacção via *Web* usando a tecnologia *Web Services* são:

- **Protocolo de transferência –** Dos vários protocolos de transferência de dados previstos pelos *Web Services* escolheu-se o protocolo HTTP para suportar o transporte das mensagens entre as várias entidades/aplicações desta arquitectura.
- **Estrutura das mensagens –** o SOAP é a especificação adoptada pelos *Web Services* responsável pela estrutura das mensagens a transmitir de forma a estas poderem ser entendidas e interpretadas por qualquer umas das aplicações distribuídas.
- **Documentos de descrição de serviços –** O WSDL é a especificação adoptada pelos *Web Services* para descrever os serviços *Web* disponíveis. No caso do presente trabalho, os documentos WSDL descrevem os serviços propostos, disponibilizados pelos **Fornecedores**.

5.3 *Broker* Proposto

O ***Broker*** proposto é uma aplicação informática (motor de busca avançado) capaz de descobrir, de uma forma automática, simples, rápida e autónoma, **Fornecedores** capazes de fornecerem um determinado **produto** às melhores condições (preço e prazo de entrega).
Para uma maior facilidade de acesso à informação existente, os serviços fornecidos pelo ***Broker*** estão disponíveis em páginas *Web* (ver Figura 29). Para aceder ao ***Broker*** basta ter um computador, PDA ou telemóvel que possua um *Web Browser* e uma ligação à Internet. Estando este serviço disponível numa plataforma *Web* encontra-se disponível em qualquer parte do mundo e a qualquer momento.

Figura 29 – Página principal do *Broker* proposto.

A aplicação ***Broker*** desenvolvida tem um conjunto de funcionalidades disponíveis para o **Cliente**, entre as quais se salientam a inscrição na Base de Dados do **Broker**, o sistema de autenticação, o sistema de recuperação de password, o sistema de envio de e-mail ao **Cliente** com informação diversa. Para além disso, a aplicação ***Broker*** providencia dois modos de funcionamento para a contratação de bens e serviços:

1. Automático (correspondente à arquitectura 5);
2. Manual (correspondente à arquitectura 4).

Passa-se agora a explicar, mais pormenorizadamente, cada um dos modos de funcionamento.

5.3.1 Modo Automático

Na situação do ***Broker*** estar em modo de funcionamento automático, todo o processo de pedir, negociar e avaliar junto dos vários **Fornecedores** o melhor orçamento para a produção de um **produto** que lhe seja solicitado por parte de um **Cliente**, é transparente para o **Cliente**. Assim, quando o **Cliente** fizer uma pesquisa por um **produto,** o ***Broker*** encarregar-se-á de realizar todo processo de contratação e o **Cliente** apenas terá como resultado a lista de **Fornecedores** que fornecem o **produto** pretendido. Neste modo de funcionamento o **Cliente** apenas diz qual o critério que quer para a selecção do **produto**, ou seja, se quer que o critério de selecção seja o preço (Preço) ou o prazo de entrega mais curto (Prazo de Entrega) (ver Figura 30).

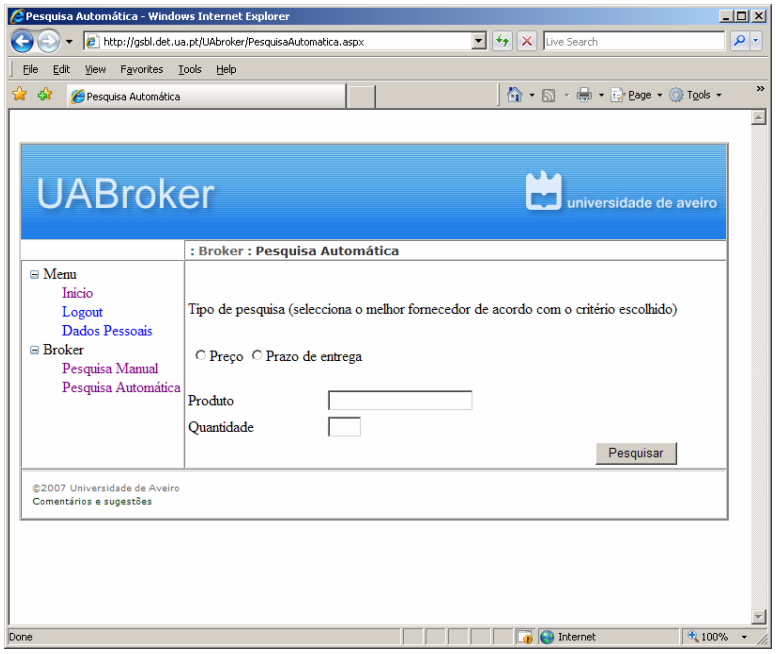

Figura 30 – Modo Automático.

Depois de seleccionado o critério (Tipo de pesquisa) pretendido, o **Cliente** tem de introduzir o **produto** que pretende e a quantidade desejada. Tendo em conta o critério escolhido, o ***Broker*** seleccionará o melhor **Fornecedor** para esse **produto** e essa quantidade. Como se mostra na Figura 31 os campos do formulário têm de ser preenchidos, caso contrário o **Cliente** não consegue finalizar o pedido.´

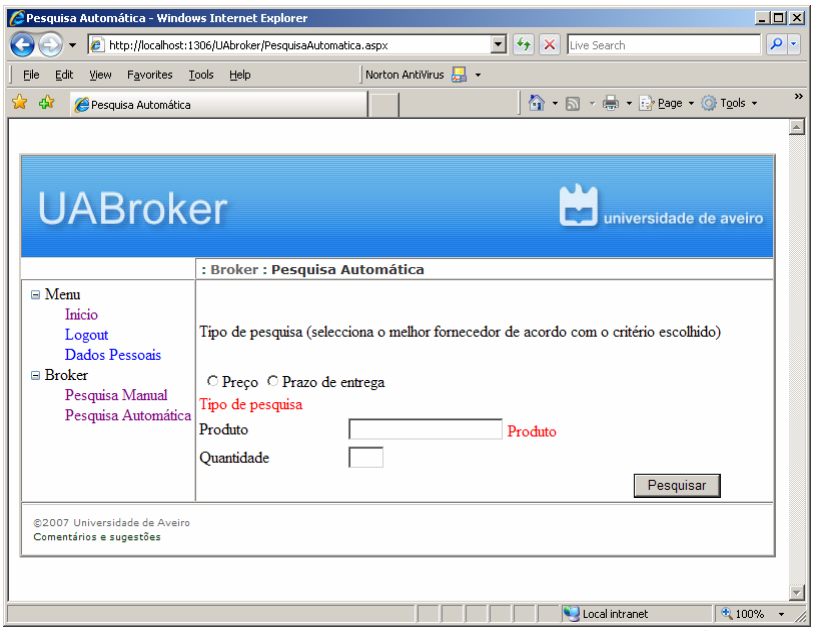

Figura 31 – Preenchimento dos campos do formulário.

Quando todos os campos da página estiverem correctamente preenchidos, como se exemplifica na Figura 32 para o caso do **Cliente** pretender comprar 25 Martelos, o ***Broker*** vai consultar a sua

lista de possíveis **Fornecedores**, contactando em seguida os serviços *Web* de cada possível **Fornecedor** (2º passo) para saber quem são os **Fornecedores** desse tipo de **produto(**Figura 33).

Figura 32 – Preenchimento correcto dos campos da página.

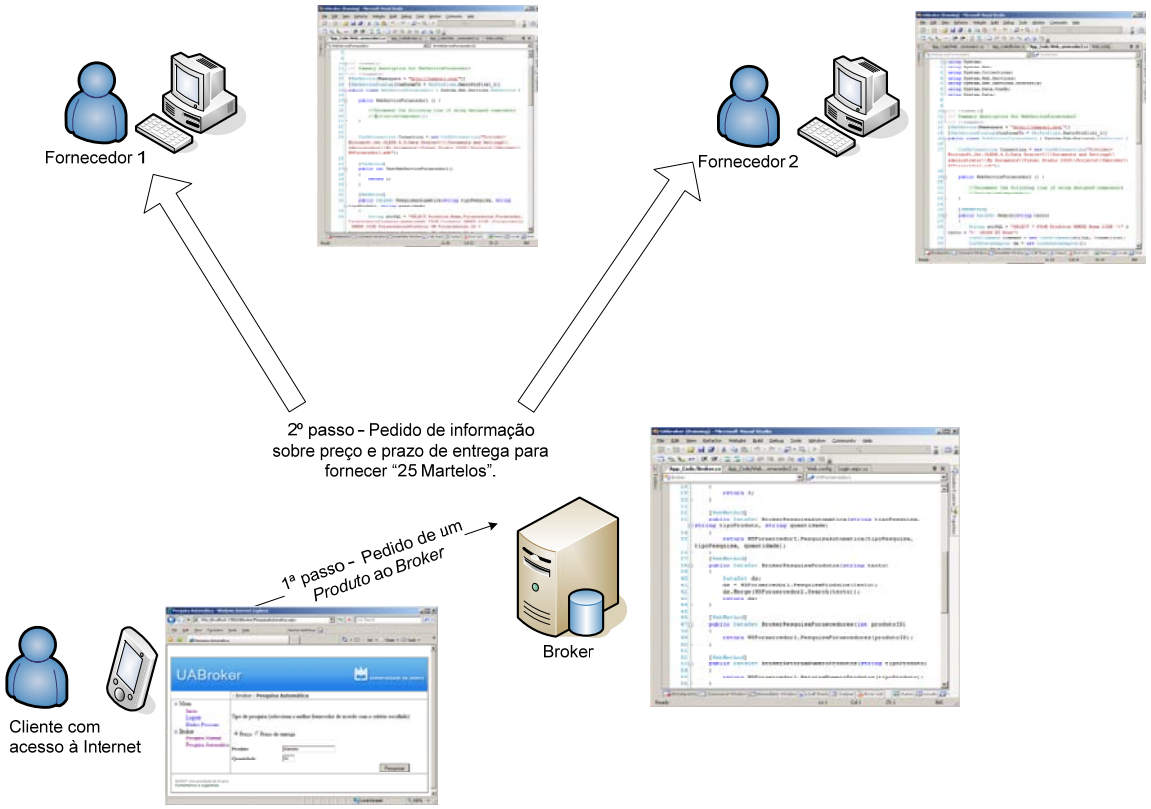

Figura 33 – Pedido de informação aos serviços *Web* de cada Fornecedor.

Os *serviços Web* de cada **Fornecedor** respondem ao pedido do ***Broker***, fornecendo a informação pretendida, o preço, o prazo de entrega e quantidade de **produto** disponível.

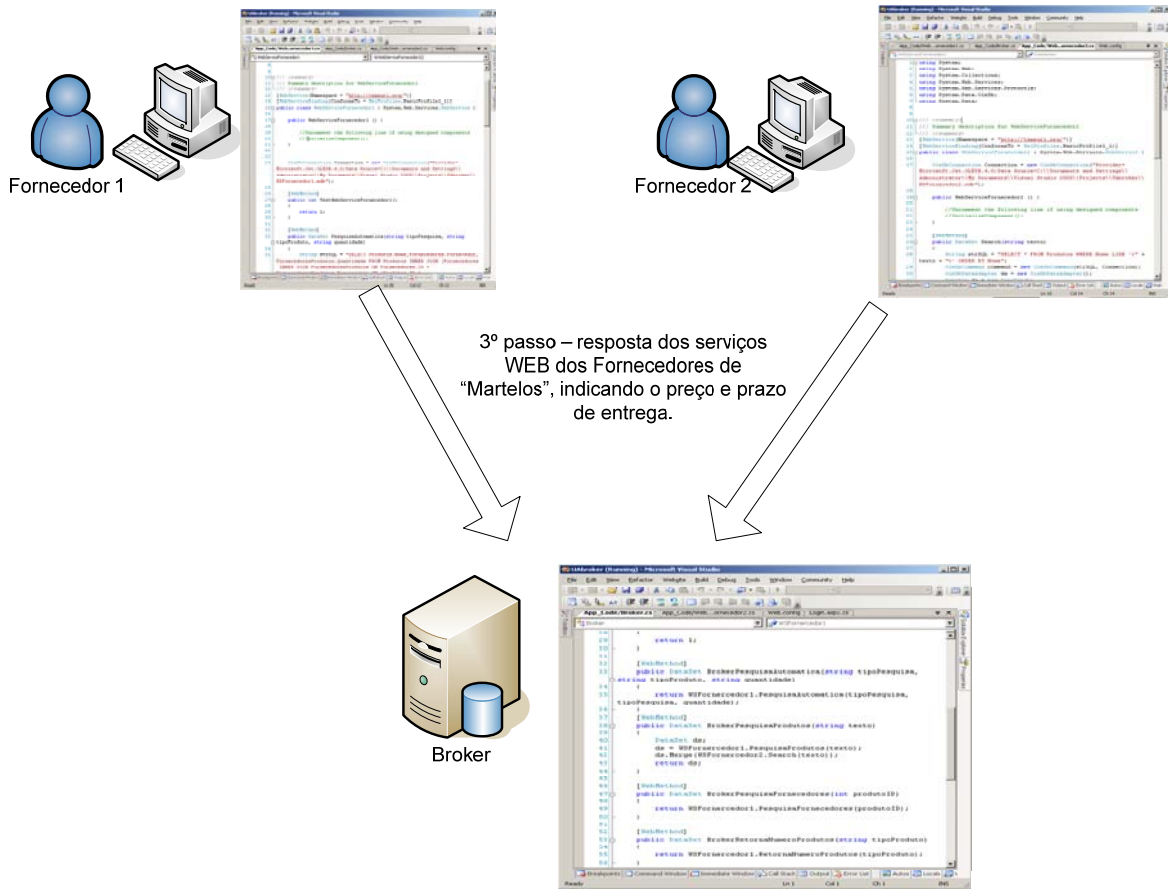

Figura 34 – Resposta dos serviços *Web* dos Fornecedores.

Em seguida o ***Broker*** selecciona o melhor **Fornecedor** de acordo com o critério escolhido pelo **Cliente**.

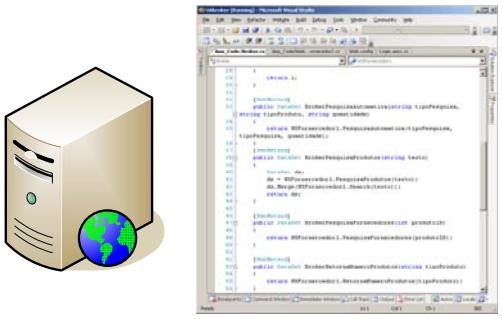

Figura 35 – Fase de selecção do melhor Fornecedor.

Depois de seleccionado o melhor **Fornecedor**, o ***Broker*** processa a encomenda e envia uma mensagem de correio electrónico para o **Cliente** a indicar que o processo de encomenda pretendido pelo **Cliente** foi processado (Figura 36).

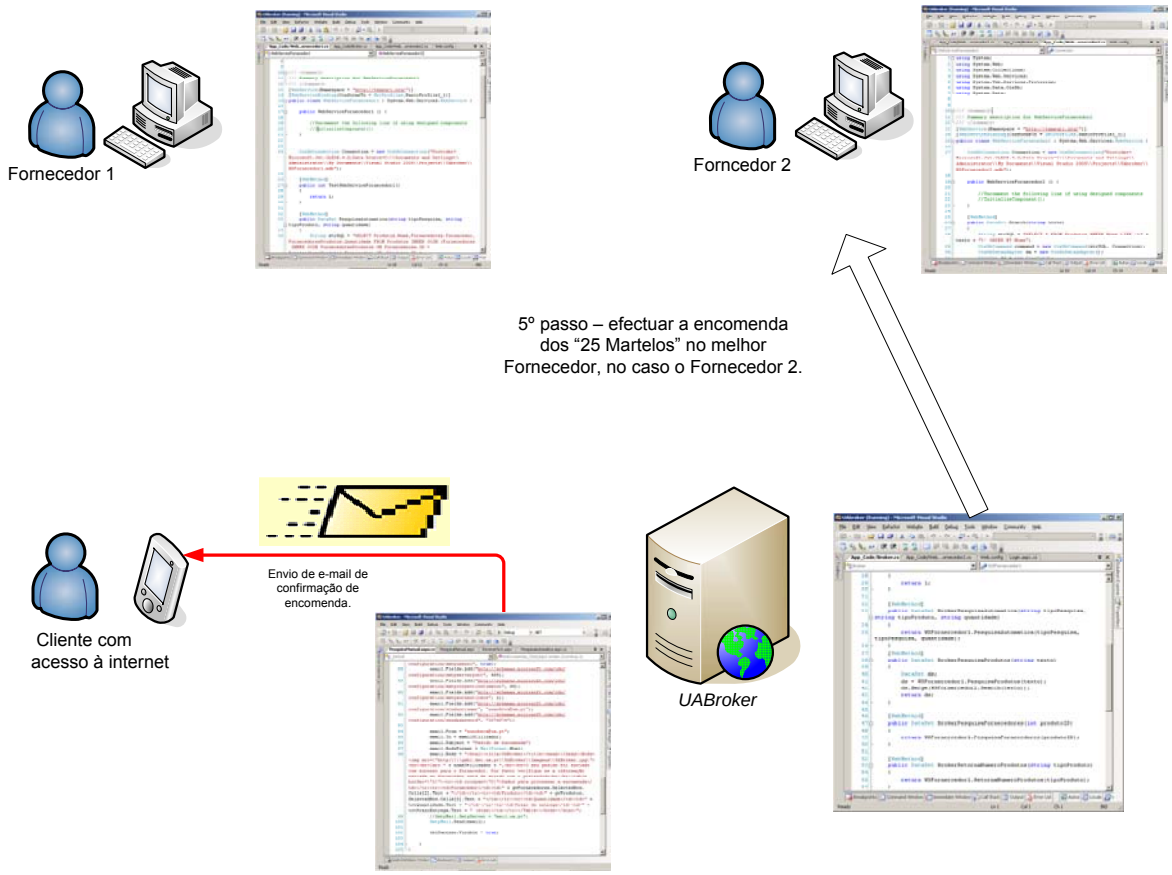

Figura 36 – Encomenda ao Fornecedor e envio de e-mail ao Cliente.

5.3.2 Modo Manual

Em modo de funcionamento manual, o ***Broker*** só tem como funcionalidade indicar ao **Cliente** os possíveis **Fornecedores** de um bem ou serviço por ele pretendido.

Por exemplo, no caso do **produto** pretendido ser "Alicates", o **Cliente,** ao fazer uma pesquisa por "Alicates", apenas terá acesso a uma listagem dos potenciais **Fornecedores** na Base de Dados capazes de fornecerem o **produto** pretendido (ver Figura 37).

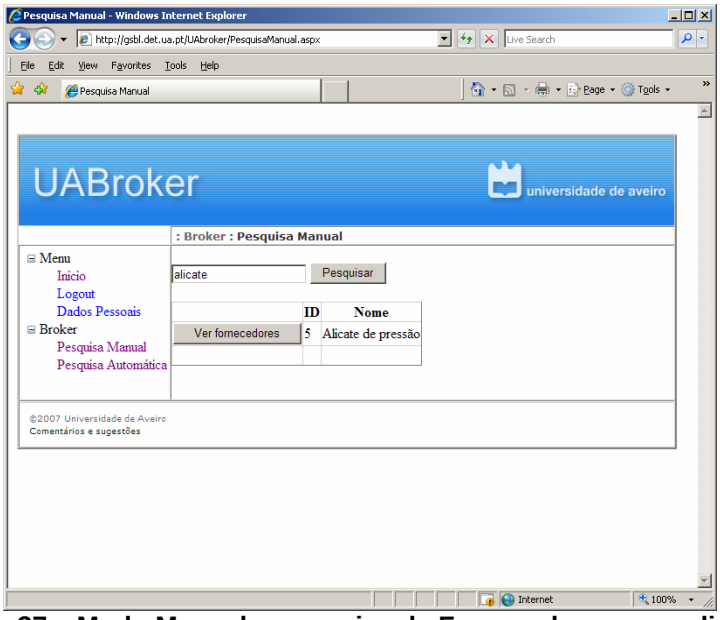

Figura 37 – Modo Manual – pesquisa de Fornecedores para alicates.

Depois de ter indicado qual o **produto** ou **serviço** pretendido, o **Cliente** tem de carregar no botão "Ver Fornecedores", que nos indica quais os **Fornecedores** existentes para fornecer "Alicates" (ver Figura 39).

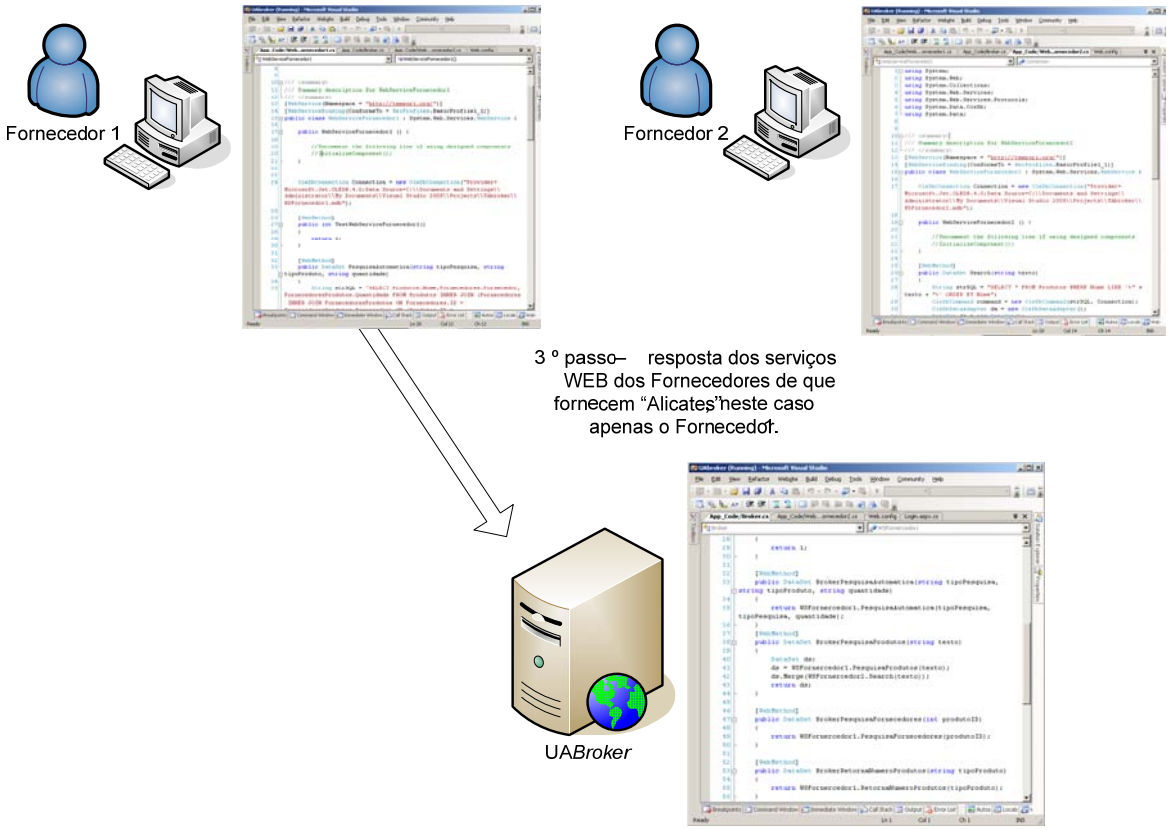

Figura 38 – O Broker quer saber quem fornece alicates.

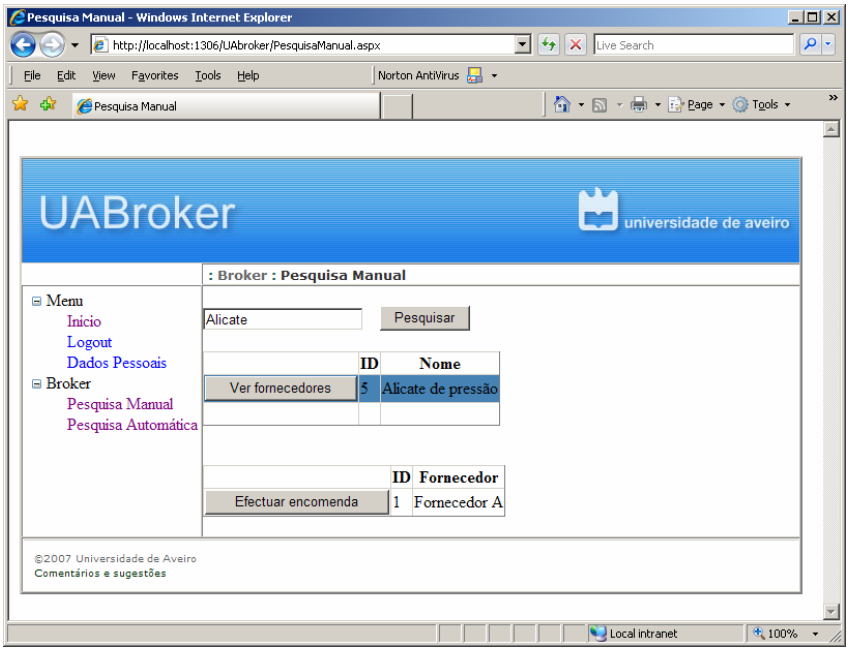

Figura 39 – Modo Manual – ver Fornecedores para alicates.

O passo seguinte é seleccionar o **Fornecedor**. Para isso o **Cliente** tem de premir o botão "Efectuar encomenda" do **Fornecedor** pretendido (Figura 40).

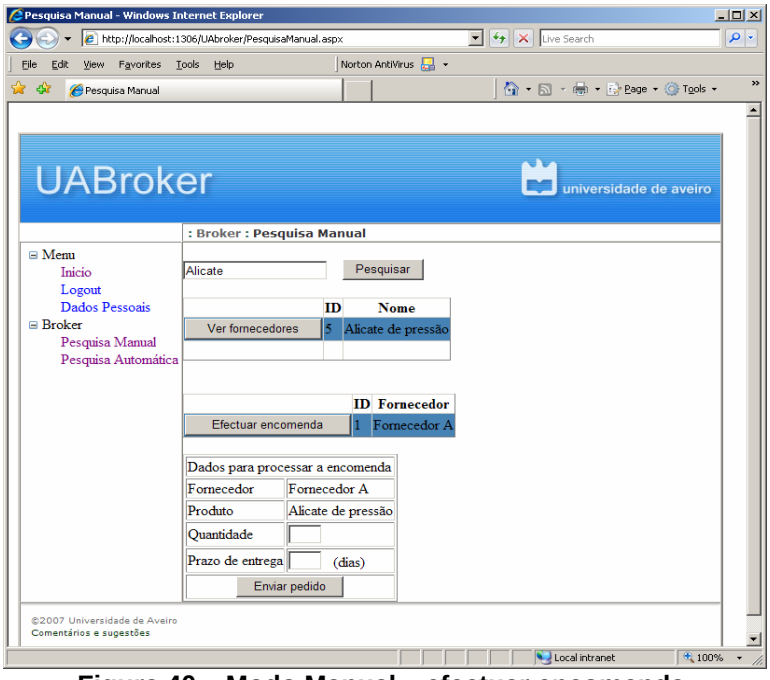

Figura 40 – Modo Manual – efectuar encomenda.

Em seguida, o **Cliente** introduz a Quantidade pretendida e o Prazo de Entrega pretendido. Posteriormente o **Cliente** vai receber as orçamentos de cada **Fornecedor**.

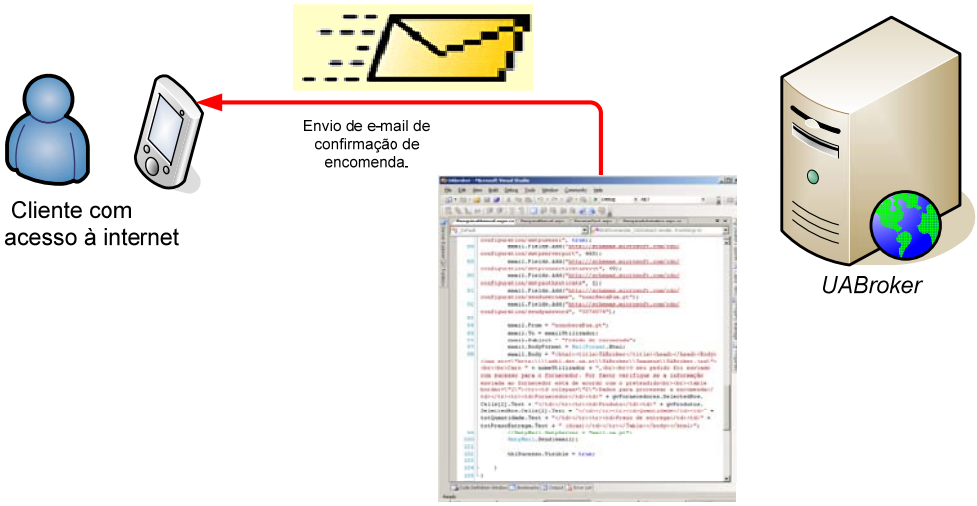

Figura 41 – Envio de um e-mail de confirmação ao *Cliente*.

5.3.3 Desenvolvimento

O ***Broker*** é uma aplicação informática desenvolvida em Visual Studio 2005, usando a linguagem de programação C#. A Base de Dados de suporte foi desenvolvida no MICROSOFT ACCESS, apesar de poder ser migrada facilmente para outros *DataBase Management Systems* tal como o SQL *Server*, MySQL. O objecto *OleDbConnection* é o objecto utilizado para estabelecer a ligação da Base de Dados à aplicação. O conjunto de páginas HTML desenvolvido permite uma melhor interacção com os **Clientes** finais, que acedem aos seus serviços através do endereço URL.

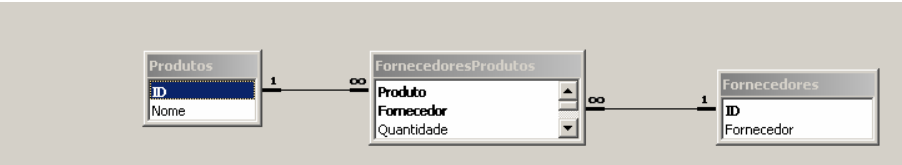

Figura 42 – Base de Dados em Microsoft ACCESS.

5.3.4 *Base de Dados do Broker* e dos Fornecedores

Na Base de Dados do ***Broker*** é onde está guardada toda a informação sobre o **Cliente**, que é usada para o processo de transacção, ou seja, inscrição do **Cliente** no ***UABroker***, esta alojada no servidor do ***Broker***. Visto a Base de Dados do ***Broker*** apenas ter a informação sobre os **Clientes**, só precisa de ter uma tabela "Utilizadores" (ver Figura 43).

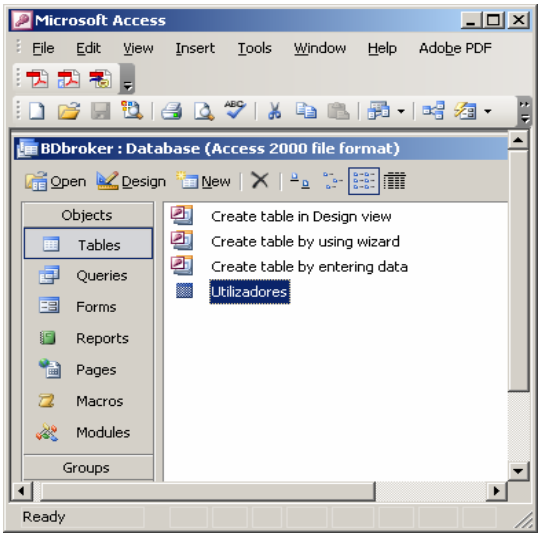

Figura 43 – Base de Dados do *Broker*.

Já os **Fornecedores** necessitam de ter uma Base de Dados devidamente estruturada, estas estão divididas em 3 tabelas. São elas as seguintes:

- **Fornecedores** – Tabela onde está a informação do **Fornecedor**;
- FornecedoresProdutos – Tabela onde está a informação dos **produtos** disponibilizados por cada **Fornecedor**;
- **Produtos** – Tabela onde está a informação do **produtos**.

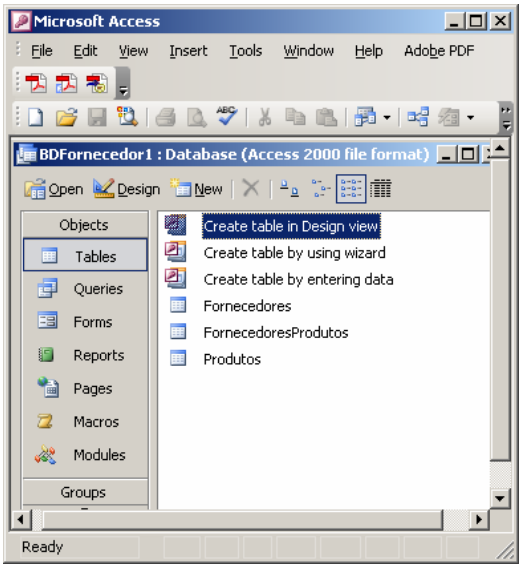

Figura 44 – Base de Dados do Fornecedor 1.

Nestas Bases de Dados é onde está armazenada toda a informação sobre o **produtos** (quantidades, preços e prazos de entrega) e informação sobre os **Fornecedores**.

5.4 Os Fornecedores

Como não foi possível usar uma instalação fabril completa para implementar e avaliar a solução proposta, foi apenas foram criados **Fornecedores** "virtuais", para assim se possibilitar o processo de pedido e negociação, bem como a escolha do orçamento economicamente e/ou temporalmente mais vantajoso. O objectivo destes **Fornecedores** virtuais é disponibilizar um *Serviço Web* que seja capaz de fornecer ao ***Broker*** a informação actualizada sobre os **produtos** ou **serviços** por si disponibilizados, desde o preço unitário e prazo de entrega para assim o ***Broker*** poder seleccionar qual o melhor **Fornecedor** para um determinado **produto**.

5.5 Serviços *Web* Propostos

Além das arquitecturas estudadas, e dos modos de funcionamento propostos nesta dissertação preverem a existência de um **Cliente**, de um ***Broker*** e de vários **Fornecedores**, são também propostos serviços *Web* que permitem que se consiga realizar uma transacção eficaz, eficiente e segura.

O ***Broker*** vai actuar como um **Cliente** destes serviços, pois vai estar a pedir informações, e a plataforma *Web* como servidor, implementando-os. Mais exactamente, quando o **Cliente** pretende um determinado serviço do **Fornecedor**, é o ***Broker*** que vai gerar as mensagens SOAP adequadas de acordo com estes serviços *Web* (WSDL). Cada mensagem SOAP identifica o nome do serviço pretendido e contém os parâmetros necessários a cada serviço. Quando a plataforma *Web* executa o serviço pedido pelo ***Broker***, o **Fornecedor** responde-lhe, enviando-lhe outra mensagem SOAP. O que se pretendeu implementar com este sistema foi que a informação disponibilizada seja o mais fidedigna possível. Para isso o nosso *Web* Service irá buscar toda a informação aos vários **Fornecedores** através dos seus *Web Services*. Assim, a informação estará sempre actualizada, salvo falhas no próprio **Fornecedor**.

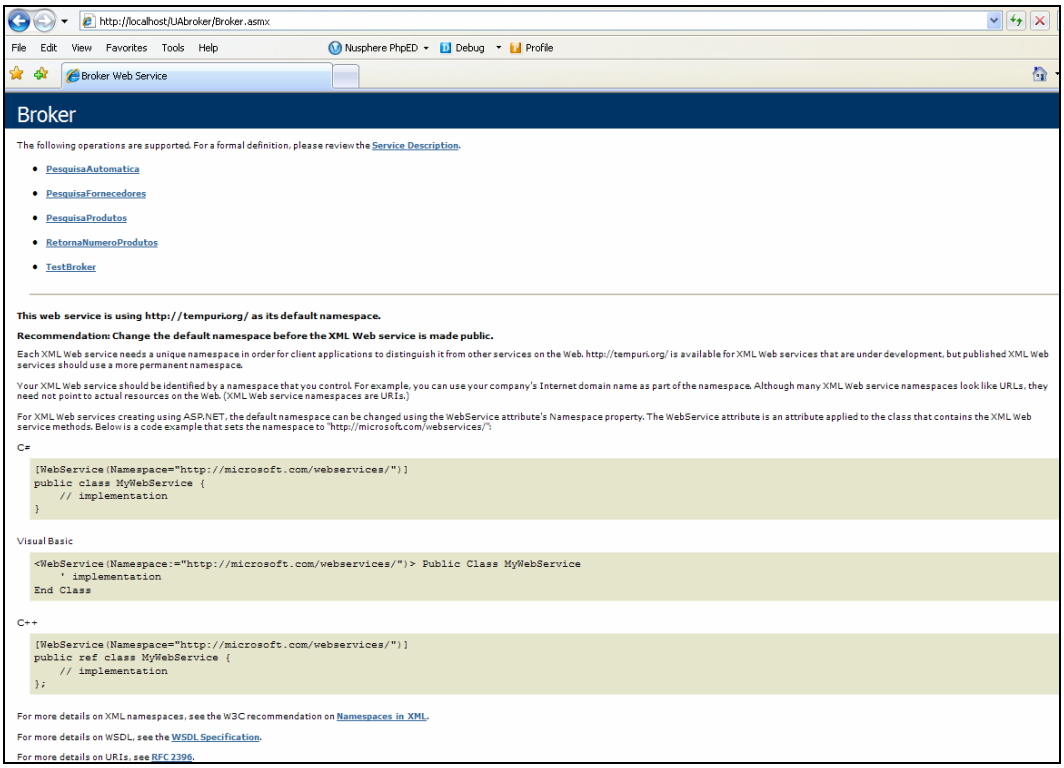

Figura 45 – UABroker.

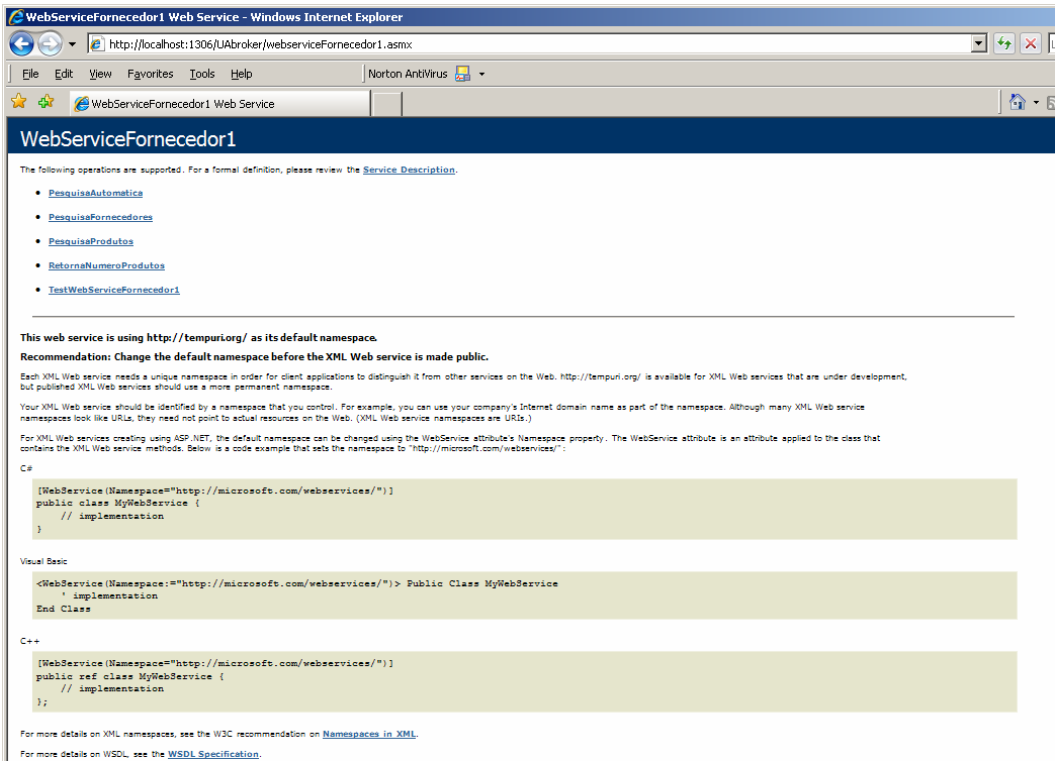

Figura 46 – *Web* Service do Fornecedor 1.

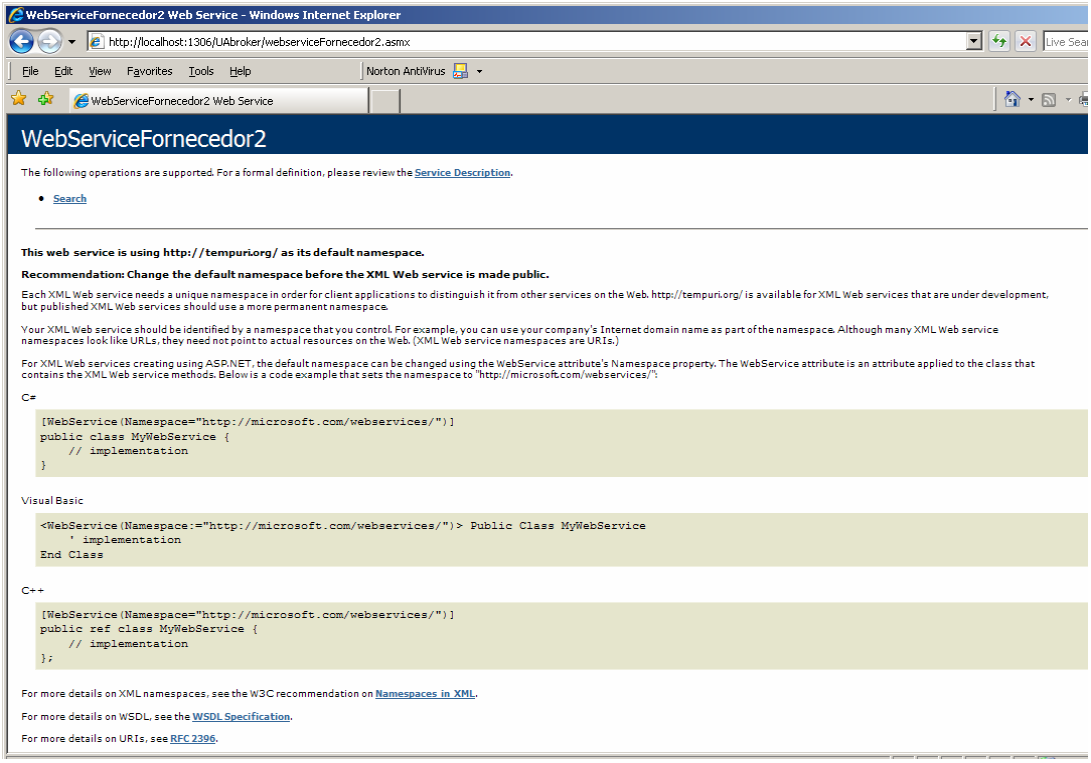

Figura 47 – *Web* Service do Fornecedor 2.

5.6 Resultados Práticos

Neste capítulo serão apresentados os resultados alcançados com a implementação da solução proposta, mais concretamente uma avaliação qualitativa das duas arquitecturas testadas, o modo manual e o automático.

Apresenta-se para tal a Tabela % onde são apresentadas as conclusões obtidas da comparação qualitativa dois modos de funcionamento tendo em conta vários critérios de comparação, sendo os critérios os seguintes. O tempo de demora da transacção desde o pedido de até a encomenda, confidencialidade dos intervenientes na transação, dependência das máquinas, complexidade do ***Broker*** e por último o número de **Fornecedores** contactados.

Tabela 5 – Tabela comparativa dos dois modos de funcionamento.

	Modo Manual	**Modo Automático**
Tempo de demora da transacção	Demorada, como é o **Cliente** a tratar de grande parte da transacção, o tempo de demora é muito dependente do factor humano.	Reduzido, como é o ***Broker*** a tratar da transacção não há dependência do factor humano.
Confidencialidade	Garantida, o **Cliente** nunca chega a saber quem são os **Fornecedores** para o **produto**.	Garantida, o **Cliente** nunca chega a saber quem são os **Fornecedores** para o **produto**.
Dependência das máquinas	Normal, visto o ***Broker*** apenas indicar quais os **Fornecedores**.	Elevada, é o ***Broker*** a tratar de toda a transacção.
Complexidade do *BROKER*	Normal, apenas indica quais os **Fornecedores** existentes para fornecer um determinado **produto**.	Elevada, precisa além de determinar os possíveis **Fornecedores**, precisa também de seleccionar a melhor **Fornecedor** para um determinado **produto**.
Nº de Fornecedores contactados	Depende do **Cliente**, podendo ser apenas um.	Todos os **Fornecedores** existentes.

CAPÍTULO 6

6 CONSIDERAÇÕES FINAIS, CONCLUSÕES E TRABALHO FUTURO

Neste capítulo, em primeiro lugar, vão ser apresentadas as considerações finais e aspectos relevantes que foram aparecendo durante realização deste estudo e elaboração desta dissertação. Em seguida será feito uma apresentação resumida do objectivo do trabalho e da solução proposta, bem como as conclusões que foram possíveis de obter. Por fim serão abordados alguns assuntos que foram abordados neste trabalho, de uma forma conceptual e genérica é certo, mas que poderiam ser alvo de futuros estudos ou evoluções, podendo enriquecer ou complementar este estudo.

6.1 Considerações Finais

Nos dias de hoje a Internet é um meio privilegiado de trocar e aceder todo o tipo de informação de uma forma rápida e eficaz. Por isso fazer um trabalho de investigação nesta área, é muito gratificante, ainda por mais tendo em conta a minha formação académica em Engenharia Mecânica.

As arquitecturas estudadas foram pensadas com o objectivo de se poder ter cada vez mais facilidade e garantia de sucesso aquando a subcontratação ou compra de **produtos** ou **serviços**. Para isso pensou-se em arquitecturas em que fosse possível a flexibilidade, dinamismo, virtualidade e agilidade na escolha dos **Fornecedores**, por exemplo, se o **Fornecedor** escolhido não pudesse cumprir com o que estava combinado e caso fosse o ***Broker*** responsável pela transacção, este deveria ser capaz de se adaptar, seleccionando outro **Fornecedor** ou mesmo fazer novo pedido de cotação tendo em conta os critérios definidos pelo **Cliente**.

O Serviço *Web*, ***UABroker*** desenvolvido é de simples utilização, flexível, disponível 24 horas por dia durante todo ano, capaz de uma forma automática e autónoma localizar os **Fornecedor**es existentes capazes de fornecer um determinado **produtos** ou **serviços**, pedir preços e seleccionar a melhor proposta tendo em conta critérios pré-definidos.

Há ainda trabalho para fazer, pois o objectivo deste trabalho era o estudo arquitecturas flexíveis, ágeis e reconfiguráveis para a subcontratação, usando um sistema, inteligente, autónomo, flexível e amigável com o utilizador.

6.2 Conclusões

Nesta dissertação foram apresentadas cinco arquitecturas para representar um sistema flexível, virtual, dinâmico de subcontratação de **serviços** ou **produtos**, das quais se concluiu que as arquitecturas que têm *Regulador de Mercado* são mais seguras visto que se garante a ideonidade das entidades participantes e que assim se aumenta a confiança em torno da transacção, que as arquitecturas em que é o ***Broker*** a tratar da transacção esta torna-se mais rápida, eficiente, flexível e ágil, visto o ***Broker*** ter inteligência para se adaptar rapidamente às situações que lhe vão aparecendo.

Um ponto importante é a confidencialidade de todos os participantes e esta é garantia nas arquitecturas nº3, 4 e 5, sendo parcialmente conseguida na arquitectura nº2 visto o C*liente* só entrar em contacto com os serviços Web do **Fornecedor** na parte do acompanhamento do estado da encomenda. No caso da arquitectura nº1 a confidencialidade não é garantida visto ser o **Cliente** a tratar de grande parte da transacção, sendo que a função do ***Broker*** é apenas fornecer a listagem dos possíveis **Fornecedores** para um determinado **produto** ou **serviço**.

Para o desenvolvimento da parte experimental desenvolveu-se uma aplicação automática e autónoma de localização e selecção de **Fornecedores** para um determinado **produto** ou **serviço**, como forma de facilitar a contratação e aquisição de um determinado **produto** ou **serviço**, melhorar as performances de produção de **produtos** e de stocks de matérias primas. Isto porque, quando uma empresa receber uma grande encomenda para a qual não tenha capacidade fabril para a cumprir, teria de adquirir maquinaria e pessoal, o que acarretava um enorme investimento, o qual poderia não ter retorno caso não houvesse mais encomendas desse género. Nestas situações as empresas passam antes a subcontratar o serviço a outras empresas, assim garante-se que mesmo que apareçam grandes encomendas, a empresa consegue cumprir com os prazos de entrega acordado com o **Cliente,** sem que para isso tenha de ter grandes quantidades de peças e matéria prima em stock, bastando para isso recorrer à subcontratação de **serviços** a outra ou outras empresas.

Na aplicação desenvolvida toda a informação apresentada ao Cliente encontra-se sempre actualizada, isto porque a aplicação recorre aos Serviços Web dos Fornecedores para disponibilizar a informação ao Cliente. Uma outra vantagem do *Broker* implementado é que permite que o Cliente tenha acesso, teoricamente, a um número ilimitado de Fornecedores porque cada *Web Service* dos Fornecedores poderá também ir contactar outros Fornecedores para serem seus Fornecedores e assim conseguir-se ter uma rede de Fornecedores que no limite é ilimitada, ver Figura 48.

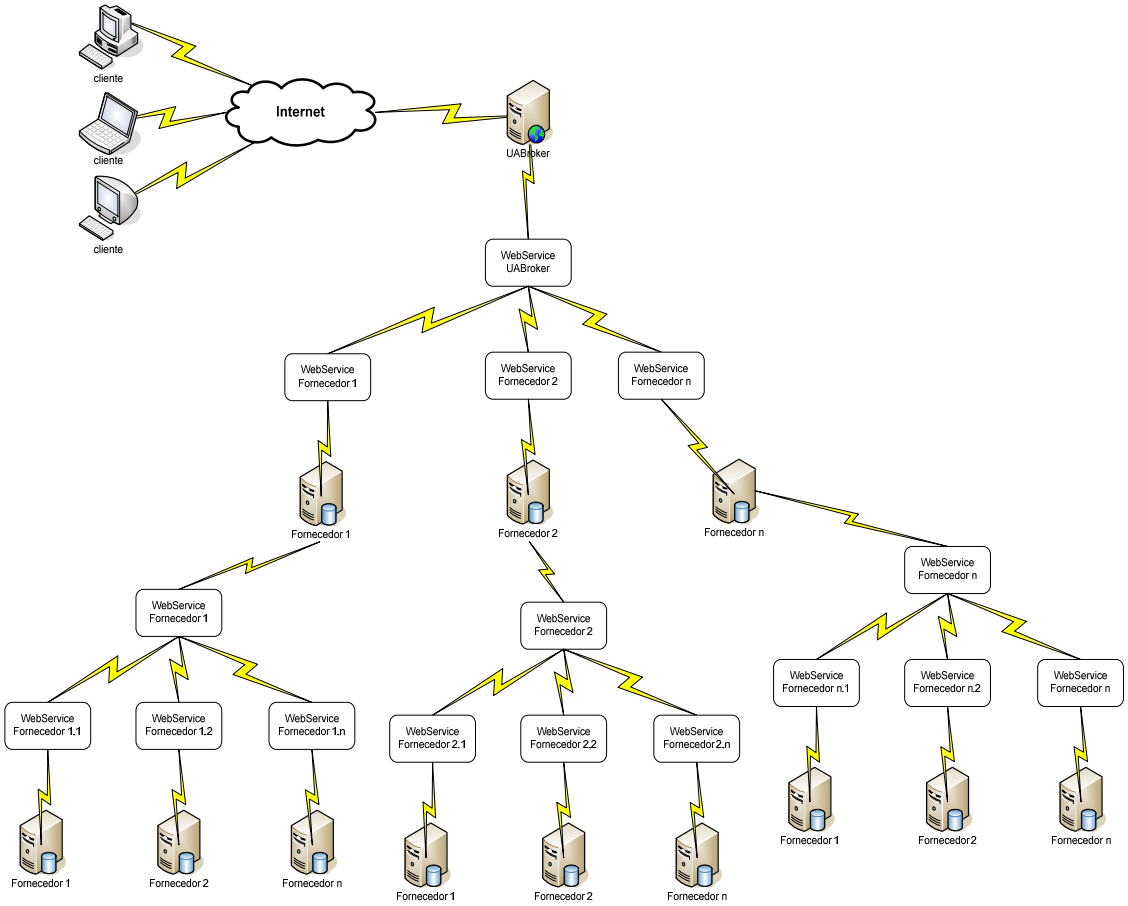

Figura 48 – Lista de Fornecedores do UABroker.

A informação sempre actualizada é garantida desde que os **Fornecedores** disponibilizem informação sobre os seus **produtos** ou **serviços** no formato de um *Web* Service, desta forma além de ser possível recorrer à informação de vários **Fornecedores**, o próprio mecanismo de pesquisa é bastante célere uma vez que toda a comunicação é baseada no protocolo SOAP, desta forma, pretendesse garantir a interoperabilidade e intercomunicação entre diferentes sistemas, através da utilização de uma linguagem (XML) e mecanismo de transporte (HTTP) padrões.

O ***Broker*** apresentado ao poder conter uma lista extensa de **Fornecedores**, faz com que o **Cliente** consiga obter um maior leque de opções para o **produto** ou **serviço** pretendido, podendo assim obter melhores condições, tanto ao nível de preço como ao nível do prazo de entrega. Para os **Fornecedores** este *Web* ***Service*** é vantajoso pelo facto de conseguirem ter uma maior visibilidade, uma vez que mostram os seus serviços a um maior número de **Clientes** e ainda pelo facto do ***Broker*** poder disponibilizar um meio de publicidade via e-mail sobre os **produtos** desses **Fornecedores**.

Outra vantagem do ***Broker*** é o facto de fazer com que seja um sistema de livre concorrência, visto o ***Broker*** analisar as propostas de acordo com critérios definidos pelo **Cliente**, conseguindo-se assim uma concorrência leal e honesta entre os vários **Fornecedores** inscritos.

À aplicação desenvolvida foi dado o nome de ***UABroker***, utiliza o protocolo HTTP e as especificações SOAP e WSDL para localizar e interagir com Serviços *Web*.

No primeiro capítulo demonstrou-se o problema da subcontratação de **produtos** ou **serviços** sem recorrer ao mecanismo proposto nesta dissertação e os custos inerentes de um incompleto, ultrapassado e demorado processo de localização e selecção de entidades **Fornecedoras** de **serviços**. Foi também apresentada uma solução genérica onde se pretendeu caracterizar os modos de funcionamento, as características conceptuais e os pontos mais importantes para uma posterior implementação.

No segundo capítulo apresentou-se a *Computação Ubíqua* e os Sistemas de Produção e Empresas flexíveis, virtuais e dinâmicas e as várias tecnologias de suporte usadas para implementar uma aplicação ***Broker***. Assim, foram descritas as tecnologias como o DCE, CORBA e Serviços *Web* (SOAP, UDDI, WSDL).

No terceiro capítulo foram apresentadas as cinco arquitecturas estudadas, explicando-se cada uma das arquitecturas em pormenor.

No quarto capítulo apresentou-se a comparação qualitativa das cinco arquitecturas estudadas e apresenta-se no final um tabela comparativa das várias arquitecturas.

No quinto capítulo apresentou-se a solução prática executada. Descreveu-se a tecnologia de suporte seleccionada, mais exactamente os *Web Services*, as entidades envolvidas e suas funcionalidades e fez-se um levantamento das caracteristicas e uma comparação qualitativa dos dois modos de funcionamento do ***UABroker*** tendo em conta o "Tempo de demora da transacção", "Confidencialidade", "Dependência das máquinas" e "Complexidade do ***Broker***" .

O aspecto mais importante do problema inicialmente apresentado, centrou-se na tentativa de reduzir os tempos, custos inerentes à localização e subcontratação de serviços ou **produtos** e arranjar soluções para se criar um sistema flexível e dinâmico para a subcontratação, garantindo assim que se consegue além de reduzir o tempo, preços dos **produtos** ou **serviços** mas também conseguir um sistema flexível para assim garantir que se consegue resolver problemas imprevistos, como seja o caso de quebra de stocks, avarias em maquinas que levam a que um **Fornecedor** não consiga cumprir os prazos e etc... . Mostrou-se que a utilização do ***Broker*** proposto torna a localização e a contratação de **produtos** ou **serviços** num processo simples, rápido e automático.

Simples, rápida e de negociação automática porque o ***Broker*** ao localizar todos os **Fornecedores** que permitem a aquisição de um determinado **produto** ou **serviço**, vai contactá-los e pedir-lhes automaticamente o seu melhor preço para o fornecimento desse **produto** ou **serviço**. Este método elimina todo o processo de realização de contactos telefónicos e de escrita de documentos para solicitar orçamentos (para cada um dos **Fornecedores** a contactar).

Por ser um processo automático, reduzem-se substancialmente os custos e os tempos de resposta dos pedidos de orçamentos aos **Fornecedores**, porque a avaliação e resposta desse orçamento não depende da disponibilidade de uma pessoa.

6.3 Trabalhos Futuros

Uma limitação do trabalho desenvolvido está no facto de não se ter simulado todas as arquitecturas propostas e assim não se ter conseguido concluir qual a melhor arquitectura para representar a subcontratação de um **produto** ou **serviço**.

Outra limitação deve-se ao facto de se ter usado na parte prática uma Base de Dados local em vez de recorrer aos servidores UDDI existentes, logo a pesquisa fica condicionada aos **Fornecedores** simulados inscritos no ***Broker***.

Uma área interessante para a evolução deste trabalho é a área da Semântica *Web*, (*Web Services* semânticos (WSS) tentam prover novas formas de processamento e raciocínio sobre a representação estática provida pelas ontologias na *Web* semântica (GÓMEZ-PEREZ et al., 2004) para assim se conseguir criar aplicações além de automáticas mas sim também com capacidade de raciocínio para que tenham capacidade de negociação mas também a capacidade de se adaptar aos imprevistos, sejam eles da responsabilidade do **Fornecedor** por não ter capacidade de cumprir o estipulado ou devido a outro tipo de falha.("...uma das maiores deficiências das soluções actuais de BI (*Business Intelligence*) é a impossibilidade de utilização da semântica do negócio para apoiar a localização, a selecção e a transformação de informação para formar e divulgar conhecimento aos seus colaboradores e parceiros estratégicos", LIEBOWITZ, 2005b)

7 REFERÊNCIAS

7.1 Artigos e Publicações

"O Modelo de *Web Services* - Como Desenvolver Aplicações em uma Nova Arquitetura de Software, *Promon Business & Technology Review Series*." [Costa, G., 2002]

"*Web Service Interface and Architecture for Accessing FieldBus System*." [Merino, 2004]

"Disponibilização e tratamento de informação na Internet orientados à interoperabilidade utilizando XML." [Gomes, A., 2005]

"Disponibilização de serviços baseados em localização via *Web Services*." [Silva, Grace ; Pereira, Patrícia e Magalhães Geovane., 2004]

"Um estudo sobre o desenvolvimento orientado a serviços." [Machado, João, 2004]

"*Web*Graf – Aplicação *Web* para Execução de Grafcets e redes de Petri em Controladores Lógicos Programáveis." [Gomes, A., 2003]

"Identificação e Controlo de Processos via Internet propõe uma arquitectura que permite." [Silva, 2003]

"Estudo e construção de um ambiente de grade computacional peer-to-peer com ênfase no balanceamento de carga." [Mattos, Érico, 2005]

"*Service-Oriented Paradigms in Industrial Automation*." [James & Smit, 2005]

"Segurança de Sistemas Ubíquos Baseada em Informações de Contexto." [Gomes, F.; Santos, F.; Goularte, R. & Moreira, E., 2004]

"*Ubiquitous communication systems for the electronics production industry: extending the CAMX framework.* "[Delamer, I.M. Lastra, J.L.M. Tuokko, R., 2004]

"*Information Technology Infrastructure and Solutions*." [Putnik, G. & Silva, J., 2005]

"*Market of Resources as a Virtual Enterprise Integration Enabler*." [Cunha, Maria Manuela. ; Putnik, Goran D. ; Gunasekaran, A. & Ávila, P., 2005]

"*BM_Virtual Enterprise Architecture Reference Model*." [Cunha, M.; Putnik, Goran D.; Silva, J. & Santos, J., 2006]

"*Reconfigurable manufacturing network: An Educational Test-Bed*." [Butala, P.; Sluga, A. & Putnik, Goran D., 2006]

"*Towards Ubiquitous Production Systems and Enterprises* (UPSE)." [G. Putnik, C. Cardeira, P. Leitão, F. Restivo, J. Santos, A. Sluga, P. Butala, 2007]

"*Web Services*: Promises and Compromises." [Arsanjani, A., Hailpern, B., Martin, J. e Tarr, P., 2003]

"*Pervasive Computing: The Mobile World.*" [Hansmann, Uwe; Hansmann, Merk Lothar Uwe; Nicklous, Martin S. e Stober,Thomas, 2006]

"*Technologies to Support the Market of Resources as an Infrastructure for Agile/Virtual Enterprise Integration.*" [Cunha, M. ; Putnik, Goran D. ; Silva, J.. ; Santos, J., 2006]

"*Developing Semantic Web Services.* "[Alesso, H. P.; Smith, C. F., 2004]

"*Market of Resources as a Virtual Enterprise Integration.*" [Cunha, M. ; Putnik, G. ; Gunasekaran, A. & Ávila, P., 2005]

"*Location- and Context-Awareness.*", Artigo apresentado na *First International Workshop*, LoCA [Strang, Thomas e Linnhoff-Popien, Claudia, 2005].

"*Location- and Context-Awareness*.", Artigo apresentado na *Second International Workshop,* LoCA [Hazas, Mike; Krumm, John; Strang, Thomas, 2006]

7.2 Dissertações de Mestrado

Alvarinhas, Hugo Miguel Gomes — "Localizar, Negociar e Contratar Serviços na *Web*." Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica. [2006]

Machado, João Coutinho — "Um estudo sobre o desenvolvimento orientado a serviços." Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós- Graduação em Informática da PUC-Rio. [2004]

7.3 Links

Microsoft MSDN [2006]
HTTP://msdn1.microsoft.com/en-us/default.aspx

Microsoft Office Access 2003, [2006]
HTTP://office.microsoft.com/en-us/assistance/CH790018001033.aspx

DCE – Distributed Computing Environment, [2006]
HTTP://www.opengroup.org/dce/info/index.htm#dcedoc

Peer-to-Peer Computing, Sousa, Nuno Miguel Tavares de, [2007]
HTTP://gnomo.fe.up.pt/~eol/MEMBERS/nuno_sousa/old/ppc/artigo.html

Simple Object Access Protocol version 1.2, [2007]
HTTP://www.ideallia"nce.org/papers/XMLe02/dx_XMLe02/papers/02-02-02/02-02-02.html

REST vs. SOAP at Amazon [2006]
HTTP://www.oreillynet.com/pub/wlg/3005?wlg=yes

E-commerce News: Amazon Opens Up E-Commerce Toolbox to Developers, [2007]
HTTP://www.ecommercetimes.com/story/39952.html

World Wide *Web* Consortium, [2007]
HTTP://www.w3.org

WSDL Tutorial [2007]
HTTP://www.w3schools.com/WSDL/default.asp

Web Services Description Language (WSDL) Version 2.0 Part 0: Primer [2007]
HTTP://www.w3.org/TR/WSDL20-primer

Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language [2007]
HTTP://www.w3.org/TR/WSDL20

Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts, [2007]
HTTP://www.w3.org/TR/WSDL20-adjuncts

IEEE [2007]
HTTP://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1417388

IBM [2007]
HTTP://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.Websphere.express.doc/info/exp/ae/cwbs_scen_stage1.html